НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

До захисту допущено
В. о. завідувача кафедри
_____ О.Л. Тимощук
«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системний аналіз і управління»
спеціальності 124 "Системний аналіз"
на тему: «Порівняльний аналіз методів прогнозування нелінійних нестаціонарних процесів»

Виконала:
Студентка IV курсу, групи КА-61
Скомороха Катерина Ігорівна _____

Керівник:
професор, д.т.н.,
Бідюк Петро Іванович _____

Консультант з економічного розділу:
доцент
Шевчук Олена Анатоліївна _____

Консультант з нормоконтролю:
доцент, к.т.н., доцент
Коваленко Анатолій Єпіфанович _____

Рецензент:
професор, д.т.н.,
Теленик Сергій Федорович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) – 6.040303

«Системний аналіз» («Системний аналіз і управління»)

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ О.Л. Тимощук

«___»_____20__ р.

ЗАВДАННЯ

на дипломну роботу студенту

Скоморосі Катерині Ігорівні

1. Тема роботи «Порівняльний аналіз методів прогнозування нелінійних нестаціонарних процесів», керівник роботи професор, д.т.н. Бідюк Петро Іванович, затверджені наказом по університету від «25»травня 2020 р. № 1143-с

2. Термін подання студентом роботи _____

3. Вихідні дані до роботи __статистичні дані стосовно розвитку вибраних_

_нелінійних нестаціонарних процесів (ННП) в економіці та фінансах_____

4. Зміст роботи __1. Аналіз актуальності задачі моделювання ННП_____

__2. Вибір типів математичних моделей для опису ННП_____

__3. Побудова СППР для виконання обчислювальних експериментів_____

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)
__1. Мета, об’єкт і предмет дослідження. 3. Постановка задачі дослідження._

__3. Деякі типи моделей, використаних для опису ННП._____

__4. Результати виконання обчислювальних експериментів._____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Шевчук О.А., доцент		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Формулювання тематики (напряму) дослідження.	03.09.2019 – 30.09.2019	Виконано
2	Аналіз актуальності задач стосовно тематики дослідження	01.10.2019 – 30.10.2019	Виконано
3	Аналіз відомих результатів стосовно тематики дослідження	01.11.2019 – 30.11.2019	Виконано
4	Формулювання задач дослідження	01.12.2019 – 30.12.2019	Виконано
5	Уточнення теми дипломної роботи	25.02.2019	Виконано
6	Збір статичних даних, попередній аналіз даних	01.03.2020 – 30.03.2020	Виконано
7	Розробка програмного продукту	01.03.2020 – 30.04.2020	Виконано
8	Виконання обчислювальних експериментів, аналіз результатів	01.05.2020 – 20.05.2020	Виконано
9	Оформлення пояснювальної записки у цілому	21.05.2020 – 31.05.2020	Виконано
10	Підготовка презентації для захисту	28.05.2020 – 29.05.2020	Виконано
11	Попередній захист дипломної роботи	29.05.2020 – 02.06.2020	Виконано
12	Захист дипломної роботи	15.06.2020 – 17.06.2020	Виконано

Студент _____ _____
(підпис) (ініціали, прізвище)

Керівник роботи _____ _____
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Дипломна робота: 123 с., 23 рис., 18 табл., 2 додатки, 10 джерел.

Об'єкт дослідження: нелінійні нестаціонарні процеси, представлені часовими рядами.

Мета роботи: побудова математичних моделей процесів в економіці та фінансах; оцінювання прогнозів; розробка програмного забезпечення для виконання обчислювальних експериментів.

Метод дослідження: математичні моделі і методи аналізу процесів в економіці та фінансах.

В роботі проведений огляд існуючих систем для аналізу часових рядів. Наведено моделі та методи прогнозування. Розглянуто і проаналізовано статистичні тести для аналізу.

Створено програмне забезпечення для моделювання та прогнозування процесів на базі авторегресійних моделей з ковзним середнім. В роботі порівняно прогнозування вибраних цін акцій за допомогою як власного програмного продукту, так і існуючого аналогу для статистичної обробки даних.

Система реалізована на базі платформи .Net Framework з використанням мови програмування C#, наведено приклади застосування програми для прогнозування реальних цін акцій компаній та процесу Лоренца. Розглянуто шляхи можливого подальшого вдосконалення системи.

НЕЛІНІЙНІ ПРОЦЕСИ, НЕСТАЦІОНАРНІ ПРОЦЕСИ, МОДЕЛЬ АВТОРЕГРЕСІЇ КОВЗНОГО СЕРЕДНЬОГО, СИСТЕМА ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ.

ABSTRACT

Bachelor thesis: 123 p., 23 fig., 18 tabl., 2 appendixes, 10 sources.

Object of research: nonlinear nonstationary processes represented by time series.

Objective: building mathematical models of economic and financial processes; evaluation of forecasts; development of software to perform computational experiments.

Method of research: mathematical models and methods of analysis economic and financial processes.

The paper reviews the most modern systems for time series analysis. Some of the known models and methods of forecasting are given. There are considered and analyzed statistical tests.

A software for analytical modeling and forecasting processes to exchange models is based on autoregressive moving average was created. The paper compares the forecasting of selected stock prices using both its own software product and the existing analogue for statistical data processing.

The system is implemented on the platform .Net Framework using the programming language C #, there are examples of programs for forecasting real stock prices and Lorenz process. There are ways of possible further improve the system.

NONLINEAR PROCESSES, NONSTATIONARY PROCESSES, AUTOREGRESSION MODELS MOVING AVERAGE, DECISION SUPPORT SYSTEMS.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

DW – Durbin-Watson (статистика Дарбіна-Уотсона);

MAPE – mean absolute percent error;

MAE – mean absolute error;

R^2 – коефіцієнт множинної детермінації;

SSE – sum of squared errors (сума квадратів похибок);

АКФ – автокореляційна функція;

АР – авторегресія;

АРІКС – авторегресія з інтегрованим ковзним середнім;

АРКС – авторегресія з ковзним середнім;

КС – ковзне середнє;

ММП – метод максимальної правдоподібності;

МНК – метод найменших квадратів;

ННП – нелінійні нестаціонарні процеси;

ПП – програмний продукт;

РМНК – рекурсивний метод найменших квадратів;

САПП – середня абсолютна похибка в процентах;

СПП – середня абсолютна похибка;

ЧАКФ – часткова автокореляційна функція.

ВСТУП

У сучасному економічному світі досить важливими є питання прогнозування динаміки процесів, оцінювання альтернативних економічних стратегій, формування бюджетів підприємств та держави та інші задачі прогнозування. Одним із найбільш популярних підходів є прогнозування на основі моделей, побудованих за експериментальними даними. Основним типом статистичних даних для прогнозування є часові ряди.

Часовий ряд – це множина рівновіддалених в часі вимірів, які характеризують поведінку процесу чи об’єкта на вибраному часовому інтервалі. Часові ряди характеризують відповідні процеси, які є стохастичними за своєю природою оскільки завжди містять випадкову складову. Таким чином, стохастична (випадкова) складова повинна завжди враховуватись у структурі моделі, що будується на основі статистичних даних.

Часові ряди є основою прогнозування та аналізу поведінки багатьох процесів. Кожний ряд можна описати своєю математичною моделлю, яка може бути використана для прогнозу конкретної змінної або побудувати одну багатовимірну модель, яка дозволить прогнозувати всі змінні одночасно. Більшість сучасних процесів (які представляються часовими рядами) в економіці, фінансах, екології та соціальних дослідженнях є нелінійними та нестаціонарними (ННП), що часто потребує значних зусиль для побудови адекватних моделей, які можуть забезпечити оцінювання високоякісних прогнозів.

Перший розділ присвячено огляду відомих рішень стосовно математичного моделювання сучасних процесів в різних галузях діяльності людини. Розглянуто основні типи процесів, що мають місце на сьогодні в економіці та фінансах. Також були розглянуті деякі існуючі комп’ютерні системи для аналізу нелінійних нестаціонарних процесів та представлені відповідні статистичні тести, що дають можливість визначити характер еволюції процесу.

У другому розділі представлено огляд математичних моделей нелінійних нестаціонарних процесів, які можуть бути використані для розв’язання задач прогнозування в економіці та фінансах.

У третьому розділі подано архітектуру розробленої системи підтримки прийняття рішень, її функціональні можливості та приклади функціональних вікон системи.

Четвертий розділ присвячено побудові математичних моделей вибраних процесів та оцінюванню короткострокових прогнозів на основі побудованих моделей.

У п’ятому розділі подано функціонально-вартісний аналіз розробленого програмного продукту.

РОЗДІЛ 1 АКТУАЛЬНІСТЬ ДОСЛІДЖЕННЯ, ОГЛЯД ВІДОМИХ РІШЕНЬ

1.1 Актуальність дослідження

Необхідність вдосконалення світової фінансової системи та методів управління фінансовими процесами є вкрай важливою в умовах сучасної економічної нестабільності. Актуальним завданням є проведення поглиблених наукових досліджень у напрямі математичного моделювання та прогнозування фінансових процесів. На українському фінансовому ринку управління фінансовими процесами залишається досить актуальним питанням, яке потребує адекватних математичних моделей.

В економіці та фінансах не існує ідеально лінійних або ідеально стаціонарних процесів, тому більшість з них є нелінійними та нестаціонарними. Такі процеси характеризуються значною кількістю складностей та особливостей, що необхідно враховувати, при моделюванні та прогнозуванні відповідних процесів. Такі процеси містять тренд або змінну дисперсію, тобто є нестаціонарними. Під трендом будемо розуміти загальну тенденцію розвитку процесу при різноспрямованому русі, яка визначена загальною спрямованістю змін показників часового ряду. Тренди вказують на довгострокові зміни процесів, а тому їх використовують для довгострокового прогнозування. Розрізняють два типи тренду: детермінований та стохастичний. Процеси з трендами та змінною дисперсією особливо характерні для фінансово-економічних процесів. Суттєвою проблемою при побудові фінансово-економічних процесів є наявність в них нелінійностей. Нелінійність означає можливість непередбачуваних змін у напрямі розвитку процесів[1].

На рисунку 1.1 зображена діаграма, яка спрощено показує сучасну класифікацію процесів в економіці та фінансах.

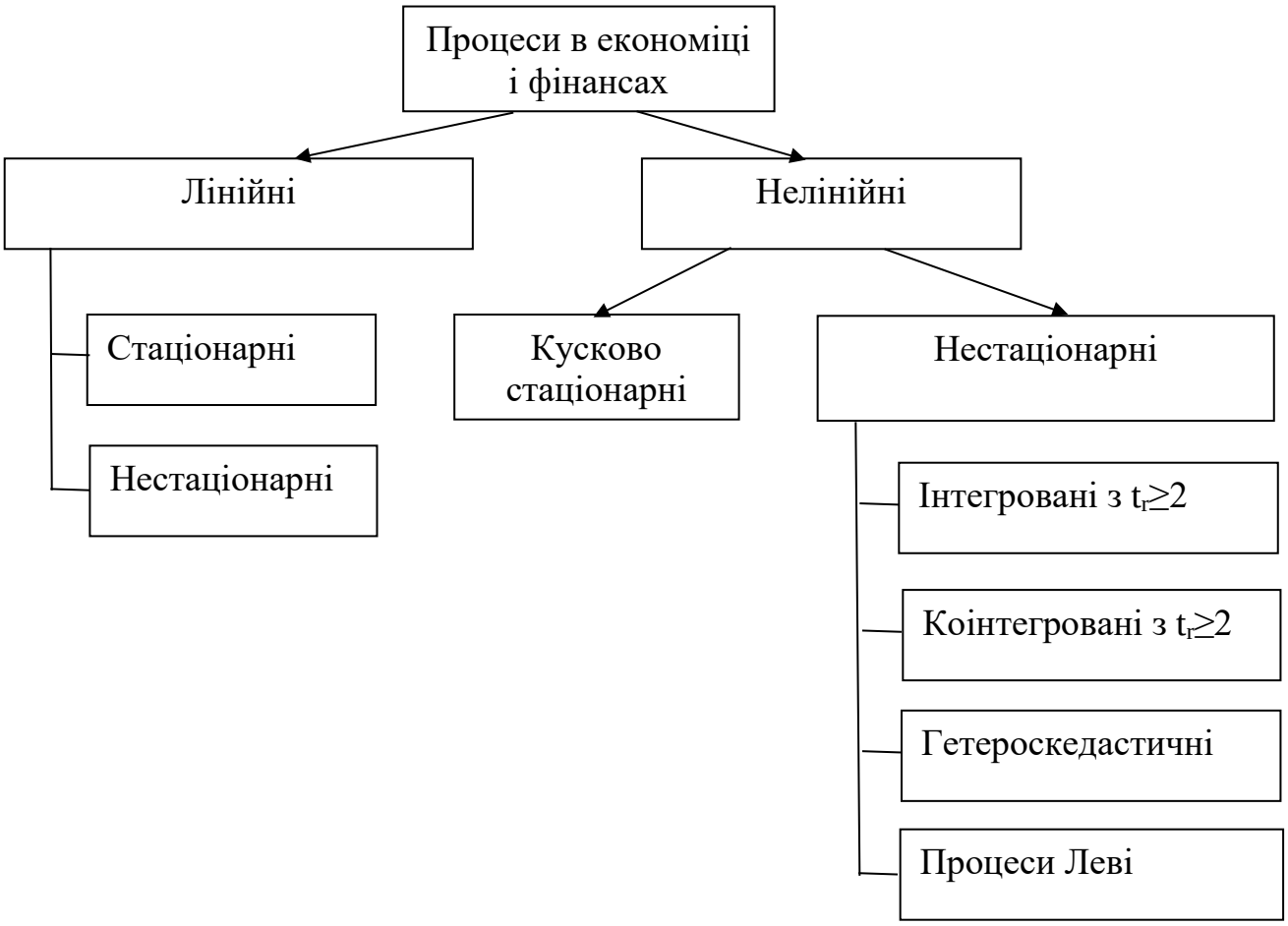


Рисунок 1.1 – Сучасні процеси в економіці та фінансах

Процес вважається лінійним, якщо він з достатнім ступенем адекватності описується лінійною моделлю і навпаки – процес вважається нелінійним, якщо він не може бути описаний з достатньою адекватністю лінійною моделлю. Лінійний процес може бути нестаціонарним, якщо він містить лінійний тренд. Серед нелінійних нестаціонарних процесів часто зустрічаються інтегровані (процеси з трендом) та коінтегровані, порядок інтегрованості яких складає 2 або більше. Поширеними у фінансах є гетероскедастичні процеси, дисперсія яких є функцією часу. Процеси такого типу є нелінійними за означенням. Процеси Леві – нелінійні нестаціонарні процеси, динаміка розвитку яких може бути дуже складною. Наприклад, вони містять різкі переходи (зміна режимів функціонування), значні викиди, зумовлені впливом випадкових факторів, значні шумові складові і т. ін.

1.2 Статистичні тести для аналізу нелінійних нестаціонарних процесів

У цьому розділі будуть розглянуті деякі методи (тести), які дають змогу визначити лінійність та стаціонарність досліджуваного процесу.

Для розв'язку задачі визначення наявності нелінійностей можна користуватися різноманітними критеріями. Одними із найпростіших методів є застосування лінійних коваріаційних функцій та дисперсійного методу. Крім розглянутих підходів можна скористатися більш простими тестами.

1.2.1 Статистика Фішера

Наприклад, статистикою Фішера:

$$\widehat{F} = \frac{\frac{1}{k-2}\sum_{i=1}^{k}\sum_{j=1}^{n_i} n_i\,(\overline{y}_i - \widehat{y}_{ij})^2}{\frac{1}{n-k}\sum_{i=1}^{k}\sum_{j=1}^{n_i}(y_{ij} - \overline{y}_i)^2},$$

де $k-$ число груп даних;

n_i- число вимірів у групі;

$\overline{y}_i$ - групове середнє;

$\widehat{y}_{\text{i}}-$ значення, що оцінюють по прямій регресії;

$n-$ загальне число вимірів.

Фактично, дана статистика являє собою відношення:

$$\widehat{F} = \frac{\text{Відхилення середніх значень від прямої регресії}}{\text{Відхилення значень } y(k) \text{ від групових середніх}}.$$

Якщо статистика $\widehat{F}$ зі

$$\nu_1 = k - 2,\ \nu_2 = n - k$$

ступенями свободи досягає або перевершує рівень значимості, то гіпотезу про лінійність потрібно відкинути.

1.2.2 Тест Дікі-Фуллера

Тест Дікі-Фуллера застосовується при визначенні наявності нестаціонарності. Суть тесту полягає в тому, що для визначення присутності одиничного кореня необхідно скористатись трьома наступними рівняннями:

$$\Delta y(k) = \gamma y(k-1) + \varepsilon(k), \quad (1.1)$$

$$\Delta y(k) = a_0 + \gamma y(k-1) + \varepsilon(k), \quad (1.2)$$

$$\Delta y(k) = a_0 + \gamma y(k-1) + a_2 k + \varepsilon(k), \quad (1.3)$$

де k – дискретний час;

γ – коефіцієнт у рівнянні

$$y(k) = a_0 + a_1 y(k-1) + \varepsilon(k).$$

Різниця між рівняннями (1.1) та (1.2), (1.3) полягає у присутності детермінованих членів a_0 і $a_2 k$ у рівняннях (1.2) і (1.3), відповідно. Рівняння (1.1) - це модель випадкового кроку, друге включає зсув у вигляді константи a_0, а третє – зсув та детермінований лінійний часовий тренд.

Досліджуємо параметр γ у трьох рівняннях. Якщо

$$\gamma = 0,$$

то послідовність $\{y(k)\}$ містить одиничний корінь. Застосування тесту Дікі-Фуллера передбачає оцінювання одного або більше з наведених вище трьох рівнянь за допомогою МНК або ММП з метою отримання оцінки параметра γ та стандартної похибки цієї оцінки. На основі оцінки та її стандартної похибки обчислюється $t-$статистика, яка порівнюється із значеннями, наведеними в таблицях Дікі-Фуллера. На основі цього порівняння приймається рішення щодо справедливості або відхилення нуль-гіпотези.

1.2.3 Графічний метод

Одним із найпростіших та наочних статистичних методів тестування процесів на наявність гетероскедастичності є графічний метод. Цей метод є найбільш доцільним, коли дослідник має недостатньо даних, необхідних для аналізу. Графічний метод застосовують тоді, коли висновки щодо наявності гетероскедастичності є суб'єктивними.

1.2.4 Тест Уайта

У тесті Уайта за допомогою звичайного методу найменших квадратів (ЗМНК) застосованого до часових рядів генеруються квадрати залишків, за якими будується допоміжна модель регресії. Регресія квадратів залишків містить константу і всі ненадлишкові регресори на множині всіх регресорів, яка включає самі регресори, їхні квадрати та взаємні добутки.

За умови висування гіпотези про наявність гетероскедастичності добуток NR^2 буде асимптотично наближуватися до розподілу хі-квадрат $\chi^2(5)$, де кількість регресорів без константи дорівнює 5 та R^2 – коефіцієнт множинної детермінації. В загальному випадку можна записати, що

$$NR^2 \leftrightarrow \chi^2(q),$$

тобто при використанні в регресії q регресорів добуток NR^2 приблизно має розподіл хі-квадрат.

Недоліком цього методу є те, що тест визначає лише присутність гетероскедастичності, та не вказує на її форму.

1.2.5 Тест Спірмана

Формула для обчислення коефіцієнта рангової кореляції Спірмана:

$$r_s = 1 - 6\left[\frac{\sum_{i=1}^{n} d_i}{n(n^2-1)}\right],$$

де d — різниця між рангами, що приписуються двом характеристикам і-го об'єкта;

n — кількість об'єктів, що ранжируються.

Приклад застосування коефіцієнту рангової кореляції для визначення гетероскедастичності наведено нижче. Нехай

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i.$$

Етап 1. Побудувати регресію для даних y та x і розрахувати відхилення ε_i .

Етап 2. Беручи абсолютні значення $|\varepsilon_i|$, ранжируємо $|\varepsilon_i|$ та x_i у зростаючому чи спадному порядку і обчислюємо коефіцієнт рангової кореляції Спірмана.

Етап 3. За f-критерієм Ст'юдента перевіряємо значимість отриманого коефіцієнта рангової кореляції.

Для перевірки значимості побудуємо t-статистику. За таблицями Ст'юдента знаходимо t. Якщо розраховане значення перевищує $t_{КП}$, це підтверджує гіпотезу про гетероскедастичність. В іншому випадку в регресійній моделі правильним є припущення про гомоскедастичність[10].

1.2.6 Тест Голдфельфа-Квандта

Тест Голдфельфа-Квандта застосовують у випадках, коли до гетероскедастичності приводить одна змінна. Нехай $\hat{\sigma}_\varepsilon^2$ позитивно корельована з i-м регресором x_i. Запишемо покрокову процедуру тестування:

а) за зростанням чи зменшенням упорядковуємо масив значень регресора x_i;

б) середні значення змінної виключаємо з аналізу c;

в) для перших та останніх значень будуємо регресії при умові, що їх достатня кількість;

г) обчислюємо значення похибок RSS_1(відноситься до регресії, яка оцінювалась за меншими значеннями x_i) і RSS_2(відноситься до регресії, яка оцінювалась за більшими значеннями x_i). За умови правдивості припущення щодо існування гетероскедастичності відношення

$$R = \frac{RSS_2}{RSS_1}$$

буде мати F-розподіл із

$$[(N - c - 2r)/(N - c - 2r)/2].$$

1.3 Існуючі системи для аналізу нелінійних нестаціонарних процесів

Використання методів прогнозування важко уявити без відповідних комп'ютерних програм. З одного боку хочеться мати найпотужнішу систему, яка може обробляти довільну інформацію, в якій запрограмовані сотні різноманітних методів аналізу. З іншого боку, складність роботи з системою має бути мінімальною.

Розглянемо порівняльний аналіз найбільш вживаних статистичних пакетів Statistica, Eviews та SAS.

Програма Statistica має модульну структуру, тобто складається з модулів, кожен з яких використовується для вирішення свого конкретного класу завдань, а саме: аналіз тимчасових рядів і прогнозування, множинна регресія, нелінійне оцінювання, факторний аналіз, моделювання структурними рівняннями, непараметрична статистика, дисперсійний аналіз, дискримінант функціональний аналіз. Декілька модулів об'єднано в групу промислова статистика: контроль якості, аналіз процесів, планування експерименту[3].

Основними компонентами системи Statistica є: електронні таблиці для введення вхідних даних, а також спеціальні таблиці виведення числових результатів аналізу; потужна графічна система для візуалізації даних і результатів статистичного аналізу; набір спеціалізованих статистичних модулів, у яких зібрано групи логічно зв’язаних між собою статистичних процедур; спеціальний інструментарій для підготовки звітів; убудовані мови програмування SCL (Statistica Command Language) і Statistica Basic, які дають змогу користувачеві розширити стандартні можливості системи.

При завантаженні пакету програм Statistica і при створенні нового файлу з'являється електронна таблиця, в якій стовпці є змінними, а рядки – спостереженнями. Зручність введення даних в програмі Statistica обумовлена тим, що файл таблиці схожий на аналогічний з програми Excel.

Дана програма дозволяє імпортувати дані з інших Windows застосувань і програм DOS, таких як: MS Excel, MS Access, Foxpro, Paradox, dbase, CSV, SPSS, а також з файлів *.txt.

На відміну від Statistica додаток Eviews не є модульною системою, проте він містить так зване вікно робочого файлу. Об'єктна структура робочого вікна дозволяє працювати одночасно з різними типами інформації. Управління об'єктами здійснюється за допомогою процедур, які у свою чергу можуть самі створювати нові об'єкти[7].

У програмі Eviews є командний рядок, який здатний проводити статистичний аналіз даних шляхом введення в нього команд. Файл з послідовністю команд можна зберегти на комп’ютер що дозволяє створювати шаблони виконаних дій.

При роботі з даними у системі Eviews, на відміну від Statistica, необхідно задати їх формат, кількість змінних і кількість спостережень. Типи даних, з якими дозволяє працювати програма: річні, піврічні, квартальні, місячні, тижневі (5 днів), тижневі (7 днів), щоденні і недатовані спостереження. У Statistica введення та редагування даних простіше, ніж в пакеті Eviews, і тому доцільніше імпортувати данні з файлу.

За допомогою пакету можна реалізувати такі основні функції:

а) введення в комп’ютер, розширення та корегування значень часових рядів або інших типів даних, наприклад, групових даних;

б) генерування нового часового ряду за допомогою математичних виразів будь-якої складності;

в) друкування графіків та діаграм різних видів;

г) обчислення коефіцієнтів регресії за допомогою методу найменших квадратів (МНК), методу максимальної правдоподібності (ММП), двокрокового МНК та нелінійного МНК[8];

д) лінійне та нелінійне оцінювання систем рівнянь;

е) одночасне оцінювання параметрів та прогнозування кількох часових рядів (векторна авторегресія);

ж) оцінювання та прогнозування гетероскедастичних процесів;

з) обчислення описової статистики часових рядів: коефіцієнти кореляції, коваріації, автокореляції, взаємної кореляції та гістограми;

и) обчислення коефіцієнтів рівнянь авторегресії та ковзного середнього (АРКС);

к) визначення лагів (величина запізнення);

л) прогнозування на основі регресії;

м) управління базою даних часових рядів;

н) запис та читання файлів даних в стандартному форматах, включаючи Excel і бази даних[3].

Широко розповсюджена у світі система SAS має високо розвинену функціональність і, як правило, забезпечує розв’язання більшості задач економетричного аналізу, моделювання і прогнозування. Компанія була заснована у 1976 р. і свою назву отримала як абревіатура від Statistical Analytical System, хоча наразі продукти компанії вийшли далеко за межі статистичного аналізу[7].

Сьогодні — це могутній комплекс з більш як двадцятьма різними програмними продуктами, об’єднаними один з одним «засобами доставки інформації». SAS є лідером щодо набору статистичних алгоритмів та потужності обчислень. До того ж система гнучка та надає користувачу можливість приєднання власних алгоритмів.

Поняття «IDS» дає зрозуміти, що її користувачеві для 100-відсоткової автоматизації діяльності фірми достатньо встановити на свій комп’ютер систему SAS — усі інші обчислювальні та організаційні функції виконує SAS/IDS.

До переваг системи SAS можна віднести можливість програмування 4GL і роботи з базами даних SQL; ділова, наукова та рекламна графіка; експертна підтримка користувачів.

SAS/IDS можна легко зв’язати з найрізноманітнішими СУБД (ADABAS, DB2, ORACLE, SQL/DS тощо) за допомогою інтерфейсів системи SAS.

До недоліків системи можна віднести: об’ємність, складність освоєння, вимоги до математичних знань користувача, високі вимоги до апаратної частини.

1.4 Висновки до розділу і постановка задачі дослідження

Розглянуто поширені типи нестаціонарних процесів в економіці та фінансах. Виявлено особливості процесів з трендом і гетероскедастичних процесів. Опрацьовано найбільш поширені тести для виявлення нелінійності та нестаціонарності процесів.

Виконано порівняння найбільш вживаних комп'ютерних систем для статистичної обробки даних. В ході порівняльного аналізу пакетів виявлені основні переваги і недоліки кожного з них. Основною перевагою пакету Statistica є простий графічний інтерфейс системи. Він не вимагає вивчення команд та розуміння сутності керування за допомогою командного рядка. Це є основним недоліком пакету Eviews. До серйозних недоліків пакету Statistica можна віднести відсутність тестів на нестаціонарність, відсутність моделювання гетероскедастичних процесів (моделі АРУГ і УАРУГ), відсутність функції моделювання процесів з трендом та моделі АРІКС.

Було сформовано основні критерії розробки власного програмного забезпечення. Потрібно створити систему зі зручним графічним інтерфейсом, введенням, редагуванням та порівняльним аналізом даних. При цьому передбачити можливість проведення тестів на нестаціонарність, моделювання гетероскедастичних процесів та процесів з трендом та командного інтерпретатора.

Постановка задачі дослідження.

1. Виконати аналіз типів сучасних процесів в економіці, фінансах, екології, соціальних дослідженнях та інших галузях людської діяльності. Вибрати процеси для виконання обчислювальних експериментів (стосовно побудови математичних моделей і обчислення оцінок прогнозів).

2. Виконати аналіз методів проектування та реалізації системи підтримки прийняття рішень (СППР) для моделювання і прогнозування динаміки фінансово-економічних процесів.

3. Спроектувати і реалізувати СППР для моделювання і прогнозування фінансово-економічних нелінійних нестаціонарних процесів.

4. Застосувати розроблену СППР до аналізу вибраних процесів – виконання обчислювальних експериментів.

4.1. Використати методи аналізу стаціонарності процесу.

4.2. На основі статистичних даних побудувати математичні моделі вибраних процесів і обчислити оцінки короткострокових прогнозів та оцінити їх якість за допомогою критеріїв.

5. Виконати порівняльний аналіз результатів виконання обчислень за допомогою власного програмного забезпечення із уже існуючими.

6. Виробити рекомендації стосовно варіантів майбутнього вдосконалення розробленої програми.

РОЗДІЛ 2 МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ НЕЛІНІЙНИХ НЕСТАЦІОНАРНИХ ПРОЦЕСІВ

2.1 Існуючі моделі

2.1.1 Моделі стаціонарних процесів

На даному етапі завданням є вибір структури моделей для реалізації у власному програмному забезпеченні. Розглянемо основні поняття структури моделі, на які будемо опиратись при виборі:

- порядок моделі (найвищий порядок рівнянь);
- вимірність (кількість рівнянь моделі);
- лаг (час запізнення по входу) та його оцінка;
- тип можливих нелінійностей;
- тип зовнішніх збурень (детерміновані і випадкові; адитивні і мультиплікативні; імпульсні і неперервні).

Розглянемо найбільш поширені моделі стаціонарних процесів:

- Модель авторегресії. Поняття авторегресії характеризує собою так звану «пам’ять» процесу. Це пояснюється тим, що поточний стан процесу в значній мірі визначається попередніми станами. Цю властивість можна продемонструвати формулою:

$$y(k) = a_0 + a_1 y(k-1) + a_2 y(k-2) + .. + a_n y(k-n).$$

Для того, щоб вирішити, чи необхідно вводити в рівняння регресії авторегресійну складову обчислюємо автокореляційну функцію змінної (АКФ) $y(k)$. За допомогою автокореляційної функції визначається порядок авторегресії.

АКФ та ЧАКФ використовують для визначення кількості затриманих в часі значень, які необхідно брати для описання процесу. Потрібно врахувати, що АКФ

та ЧАКФ дають попередню оцінку порядку авторегресійної частини. ЧАКФ моделі дає більш «чітку» оцінку порядку процесу.

- Модель ковзного середнього. Показує тенденцію зміни цін і згладжує їх несуттєві коливання. Найбільш широко застосовуються прості, зважені та експонентні ковзні середні. Наведемо формулу для кожної точки лінії графіка простого ковзного середнього:

$$MA(k) = \frac{\sum_{i=1}^{N} y(k-i+1)}{N}.$$

Не важко помітити, що у випадку простого ковзного середнього всі вагові коефіцієнти мають однакову вагу (одиничну).

Формула для обчислення зваженого ковзного середнього:

$$MA(k) = \frac{\sum_{i=1}^{N} w_i \cdot y(k-i+1)}{\sum_{i=1}^{N} w_i},$$

де N – розмір вікна ковзного середнього;

w_i – вагові коефіцієнти;

y – часовий ряд вхідних даних.

Вагові коефіцієнти у випадку експоненційного ковзного середнього розподілені за експоненційним законом. Формула для експоненційного ковзного середнього:

$$EMA(k) = \frac{\sum_{i=1}^{N} w_i \cdot y(k-i+1)}{\sum_{i=1}^{N} w_i} = \frac{w_1 \cdot y(k) + w_2 \cdot y(k-1) + \ldots + w_n \cdot y(k-n)}{\sum_{i=1}^{N} w_i}.$$

Після знаходження ковзного середнього модель КС виражається за наступною формулою:

$$y(k) = \sum_{j=0}^{q} b_j \varepsilon(k-j),$$

де $\varepsilon(k)$ – білий шум;

b_j – параметри моделі.

- Модель авторегресії з ковзним середнім. Дана модель АРКС(p, q) є узагальненням та комбінацією двох простіших моделей часових рядів - моделі авторегресії (АР) та ковзного середнього (КС). Запишемо це за допомогою формули:

$$y(k) = a_0 + \sum_{i=0}^{p} a_i y(k-i) + \sum_{j=0}^{q} b_j \varepsilon(k-j),$$

де $\varepsilon(k)$ – білий шум;

a_i і b_j – коефіцієнти авторегресії та ковзного середнього відповідно.

Основною перевагою АРКС-процесів є менша кількість параметрів ніж у АР- та КМ-процесів.

Існує два підходи до побудови АРКС.

Підхід 1. За залишками АР(p) рівняння моделі:

а) Визначення p – порядку авто регресійної складової.

1) Обчислити АКФ та ЧАКФ для вихідної змінної.

2) За отриманими значеннями АКФ та ЧАКФ визначити p – порядок авто регресійної складової.

б) Визначення q – порядку КС.

1) Обчислити коефіцієнти моделі АР(p).

2) Обчислити коефіцієнти АКФ та ЧАКФ залишків моделі АР(p).

3) За отриманими значеннями АКФ та ЧАКФ визначте q.

Підхід 2. За вихідним сигналом y :

а) Побудова КС за вихідним сигналом *y*.

б) Визначення q – порядку КС. Обчислити коефіцієнти АКФ та ЧАКФ побудованого КС.

в) Оцінювання коефіцієнтів $a_0,...,a_p,b_1,...,b_q$ АРКС(p, q).

Умови $\sum_{i=1}^{p} a_i < 1$, та $\sum_{j=1}^{q} b_j \to 1$ на практиці не завжди виконуються. У кожному індивідуальному випадку потрібно проаналізувати результати прогнозування на декілька кроків. Якщо прогнозування дає хороші результати, то модель можна вважати прийнятною незважаючи на невиконання умов[2].

2.1.2 Модель авторегресії з інтегрованим ковзним середнім

За характером зміни в часі тренд може бути детермінованим або стохастичним. Для опису детермінованого тренду часто використовують поліноми від часу вигляду:

$$y(k) = a_0 + a_1 \cdot k + a_2 \cdot k^2 + ... + a_m \cdot k^m + \varepsilon(k),$$

де k – дискретний час;

$e(k)$ – похибка моделі.

Послідовність $\{e(k)\}$ буде містити всі коливання, що накладаються на тренд.

В результаті видалення тренду з процесу модель буде складатися з рівняння для тренду і АР(p) або АРКС(p, q) рівняння для опису коливань, що накладаються на тренд.

Для видалення тренду першого порядку (лінійний тренд) потрібно взяти перші різниці, для квадратичного тренду – другі різниці і так далі.

Модель авторегресії з інтегрованим ковзним середнім можна зробити стаціонарною взяттям різниць деякого порядку від вихідного часового ряду. АРІКС(p, d, q) означає, що різниці часового ряду порядку d підпорядковуються моделі АРКС(p, q).

Формула моделі АРІКС(p, d, q):

$$\Delta^d y(k) = a_0 + \sum_{i=0}^{p} a_i \Delta^d y(k-i) + \sum_{j=0}^{q} b_j \varepsilon(k-j),$$

де $\varepsilon(k)$ - білий шум;

a_i і b_j – коефіцієнти авторегресії та ковзного середнього відповідно;

Δ^d – оператор різниці часового ряду порядку d.

Зазначимо, що при d = 0 отримуємо АРКС-модель.

2.2 Методика побудови моделі нелінійних нестаціонарних процесів

Для реалізації СППР для прогнозування часових рядів вибираємо методику на основі застосування методів структуризації задач. Схема методики побудови моделей та прогнозування на основі часових рядів зображена на рисунку 2.1.

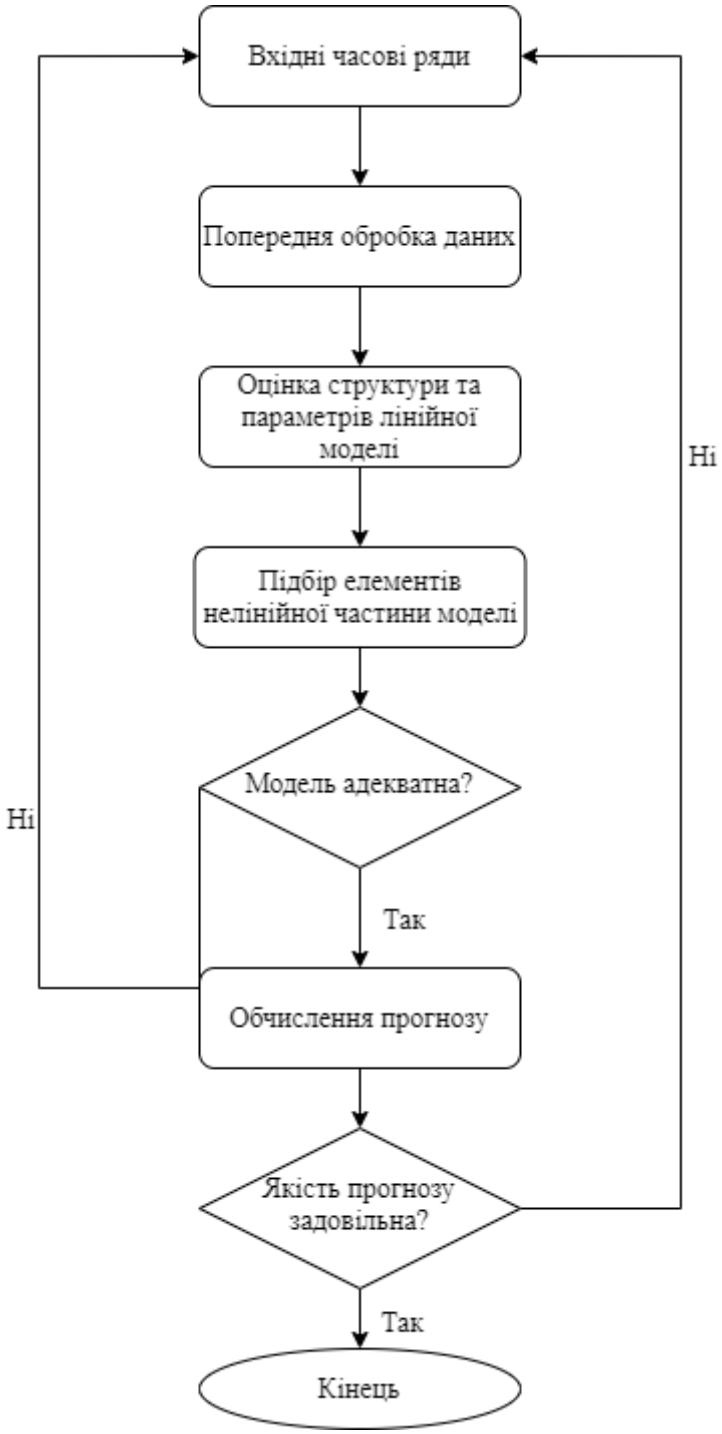


Рисунок 2.1 – Методика побудови моделі

Розглянемо детальніше кожен крок.

Крок 1. Попередня обробка:

- заповнення пропусків;
- згладжування екстремальних значень;
- логарифмування;

- нормування в діапазоні від –1 до +1;
- диференціювання.

Крок 2. Оцінка структури та параметрів лінійної моделі. Для перевірки наявності нелінійності використовуються тест Фішера та кореляційні функції вищих порядків.

Крок 3. Підбір елементів нелінійної частини моделі. Для перевірки на гетероскедастичність застосовуємо тест Уайта. Для перевірки на наявність тренду використовуємо тест Дікі-Фуллера.

Крок 4. Перевірка моделі на адекватність. За допомогою критеріїв адекватності моделі робимо висновок про якість та доцільність використання побудованої моделі. Також оцінити якість моделі можна виконавши прогноз на декілька кроків. Якщо модель не задовільна повертаємося на крок 1.

Крок 5. Побудова та оцінка якості прогнозу. Після обрання моделі переходимо до прогнозування. Обчислюємо прогноз поведінки ряду та оцінюємо точність прогнозу. Оцінити якість прогнозу можна за допомогою критеріїв вибору кращого прогнозу.

Якщо точність прогнозу не задовільна переходимо до кроку 1. Інакше, завершуємо дослідження[2].

2.3 Критеріальна база для оцінювання адекватності моделей і якості прогнозів

2.3.1 Критерії адекватності моделей

Для того, щоб оцінити значущість коефіцієнтів математичної моделі в статистичному сенсі і дослідити корельованість між значеннями похибки моделі (вони мають бути не корельованими) будемо застосовувати критерії адекватності моделі. Розглянемо статистичні параметри, які входять в множину даних критеріїв:

- t-статистика Стьюдента. За допомогою t-статистики визначають значимість коефіцієнтів регресії. Формула для обчислення:

$$t = \frac{\hat{a} - a^0}{SE_{\hat{a}}},$$

де $\hat{a}$ – оцінка коефіцієнта моделі;

a^0 – початкова гіпотеза;

$SE_{\hat{a}}$ – стандартна похибка оцінки.

В якості початкової гіпотези щодо значимості оцінки пропонується висувати гіпотезу, протилежну бажаному результату. В нашому випадку необхідно висувати початкову гіпотезу, що коефіцієнт незначимий. Даний вибір спрощує розрахунки та дає можливість коректно підійти до визначення значимості оцінок.

- Коефіцієнт детермінації R^2. Визначається відношенням дисперсії тієї частини часового ряду основної змінної, що описується отриманим рівнянням, до вибіркової дисперсії цієї змінної. Формула для обчислення:

$$R^2 = \frac{\mathrm{var}(\hat{y})}{\mathrm{var}(y)} = 1 - \frac{SSE}{SST},$$

де $\mathrm{var}(\hat{y})$ – дисперсія залежної змінної, оціненої за допомогою побудованої моделі;

$\mathrm{var}(y)$ – дисперсія вимірів залежної змінної;

$SSE = \sum_{k=1}^{N} [y(k) - \hat{y}(k)]^2$ – сума квадратів похибок моделі;

$SST = \sum_{k=1}^{N} [y(k) - \bar{y}]^2$ – загальна сума квадратів;

$\bar{y}$ – середнє значення;

$SST = SSE + SSR$.

При $R^2 = 1$ дисперсії вимірів оціненої за рівнянням змінної та цієї ж змінної збігаються. Коефіцієнт детермінації трактується також як міра інформативності моделі, якщо дисперсія є мірою інформативності.

- Сума квадратів помилок моделі $\sum e^2(k)$, тобто

$$SSE = \sum_{k=1}^{N} [\hat{y}(k) - y(k)]^2,$$

де $\hat{y}(k) = \hat{a}_0 + \hat{a}_1 \hat{y}(k-1) + \hat{a}_2 \hat{y}(k-2) + \hat{b}_1 x(k) + b_2 z(k)$;

$y(k) -$ вимірювання;

$N -$ довжина вибірки.

Найкращою, за цим критерієм, буде та модель, для якої значення $\sum e^2(k)$ є мінімальним.

- Інформаційний критерій Акайке (AIC). Цей критерій враховує суму квадратів похибок, кількість вимірів N і кількість оцінюваних параметрів моделі p. Формула для обчислення:

$$AIC = N \ln\left(\sum_{k=1}^{N} e^2(k) \right) + 2n.$$

Так як критерій залежить від суми квадратів похибок, то модель буде кращою при найменших значеннях. Даний критерій є більш інформативним ніж критерій суми квадратів похибок, оскільки додатково враховує довжину вибірки і кількість параметрів.

- Статистика Дарбіна-Уотсона (Durbin-Watson). Формула для обчислення:

$$DW = 2 - 2\rho,$$

де $\rho = E[e(k)\,e(k-1)]/\sigma_e^2$ – коефіцієнт кореляції між сусідніми значеннями похибки;

σ_e^2 – дисперсія послідовності похибок $\{e(k)\}$.

Найкраще значення критерію досягається при повній відсутності кореляції між похибками і дорівнює 2. Мінімальним та максимальним значенням може бути 0 (при $\rho = 1$) та 4 (при $\rho = -1$) відповідно.

- Статистика Фішера F. Визначає ступінь адекватності моделі та пропорційна відношенню:

$$F \sim \frac{R^2}{1 - R^2},$$

де R^2 – коефіцієнт детермінації.

Не важко помітити, що $R^2 \to 1$, то $F \to \infty$, тому модель буде більш адекватною при більшому значенні F.

- Коефіцієнт Тейла. Формула для обчислення:

$$U = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i)^2} + \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i)^2}}.$$

Величина коефіцієнта Тейла знаходиться в діапазоні між 0 і 1. Модель можна назвати ідеальною при значенні 0. Це означає, що прогнозовані значення співпадають з реальними. При U=1 ряди є не корельованими і така модель не може використовуватися для прогнозу[9].

2.3.2 Критерії вибору кращого прогнозу

Для того, щоб оцінити якість моделі, потрібно визначити наскільки точно вона відтворює реальні часові ряди. Формальними критеріями оцінки якості прогнозу є формальні статистики, точки перегину, чутливість до зміни початкових даних та чутливість до зміни коефіцієнтів.

1. Середньоквадратична похибка (СКП):

$$СКП = \sqrt{\frac{1}{S}\sum_{i=1}^{S}\left(y(k+s) - \hat{y}(k+s,k)\right)^2}\,.$$

2. Середня похибка (СП):

$$СП = \frac{1}{S}\sum_{i=1}^{S} y(k+s) - \hat{y}(k+s,k)\,.$$

3. Середня похибка у відсотках:

$$СПП = \frac{1}{S}\sum_{i=1}^{S}\frac{y(k+s) - \hat{y}(k+s,k)}{y(k+s)} \times 100\%\,.$$

4. Абсолютна середня похибка у відсотках:

$$АСПП = \frac{1}{S}\sum_{i=1}^{S}\frac{\left|y(k+s) - \hat{y}(k+s,k)\right|}{\left|y(k+s)\right|} \times 100\%\,.$$

5. Максимальна абсолютна похибка (МАП):

$$МАП = \max\left\{\left|y(k+1) - \hat{y}(k+1,k)\right|, \cdots, \left|y(k+s) - \hat{y}(k+s,k)\right|\right\}.$$

6. Мінімальна абсолютна похибка (МіАП):

$$МіАП = \min\left\{\left|y(k+1) - \hat{y}(k+1,k)\right|, \cdots, \left|y(k+s) - \hat{y}(k+s,k)\right|\right\}.$$

Оцінювання моделей за точками перегину є важливим показником, та формульного тесту даних властивостей сьогодні не існує. Перевірити чи включає модель точки перегину можна завдяки візуальній оцінці реальних і прогнозованих рядів. Також одним з найважливіших тестів якості моделі є аналіз чутливості до початкових даних. Якісною вважається модель, результати прогнозування якої незалежні від початкових даних[6].

2.4 Висновки до розділу

У цьому розділі зроблено огляд математичних моделей для опису нелінійних нестаціонарних процесів та їх характеристик. Розглянуто найбільш поширені та чітко аргументовані моделі, за якими можна побудувати якісний прогноз.

Для власного програмного забезпечення було обрано моделі АР, АРКС та АРІКС. Дані моделі мають теоретично обґрунтований алгоритм, що є перевагою для програмної реалізації. Недоліком методів є складність в описанні статистичних методів вибору порядку моделі.

Подана узагальнена методика побудови моделі нелінійних нестаціонарних процесів та розглянуті основні її етапи.

Опрацьовано критеріальну базу перевірки моделі на адекватність та вибору найкращої моделі за якістю прогнозу.

РОЗДІЛ 3 СИСТЕМА ПІДТРИМКИ ТА ПРИЙНЯТТЯ РІШЕНЬ ДЛЯ МОДЕЛЮВАННЯ І ПРОГНОЗУВАННЯ НЕЛІНІЙНИХ НЕСТАЦІОНАРНИХ ПРОЦЕСІВ

3.1 Архітектура СППР

Система підтримки прийняття рішень або СППР — це комп'ютерна система, яка за допомогою часових рядів та їх аналізу може впливати на процес прийняття рішень в бізнесі. Дані системи надають можливість отримання корисної інформації з першоджерел, її аналізу, а також опрацювання існуючих моделей для вирішення конкретних завдань.

Архітектура створеної СППР налічує такі рівні:

- завантаження та обробка даних;
- аналіз;
- побудова моделі;
- прогноз.

На першому рівні користувач може завантажити дані із текстового файлу або ввести дані вручну. Для усунення надлишковості даних можливе їх редагування, перетворення та підготовка до аналізу;

На другому рівні проводиться візуальна оцінка даних, побудова графіків, проведення статистичного та кореляційного аналізу.

На третьому рівні будуються моделі авто регресії ковзного середнього та оцінюються параметри якості моделей, обирається краща модель.

На четвертому рівні проводиться статичне та динамічне прогнозування за створеною АРКС моделлю.

Рівні архітектури СППР наведені на рисунку 3.1.

Рисунок 3.1 – Архітектурні рівні СППР

3.2 Діаграма класів. Опис основних класів програми

Для того, щоб відобразити взаємозв’язки між компонентами програми та описати внутрішню структуру і типи відношень потрібно зобразити програму в термінології класів ООП. На рисунку 3.2 зображена діаграма класів СППР:

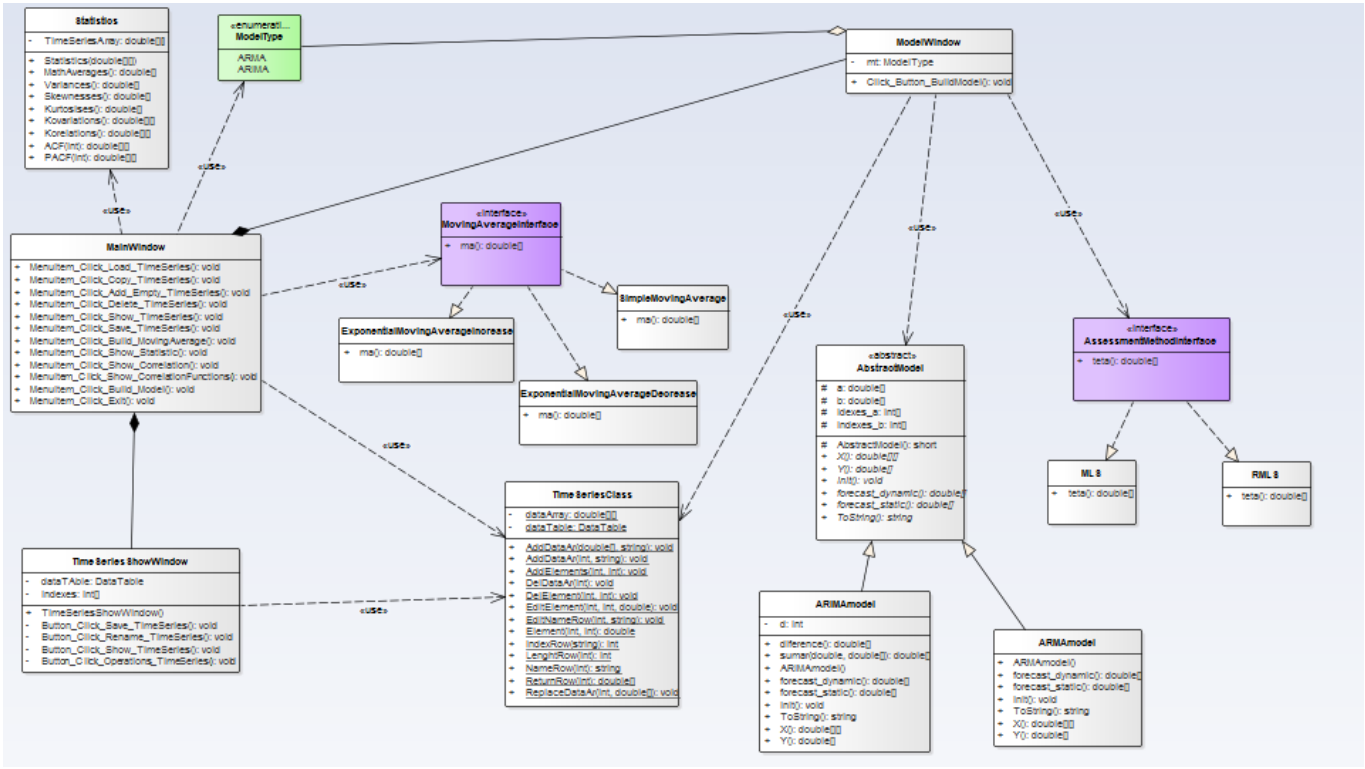


Рисунок 3.2 – Діаграма класів

Головне вікно програми реалізує клас MainWindow, а клас TimeSeriesClass є накопичувачем даних в процесі роботи програми. Розглянемо детальніше дані класи.

Клас TimeSeriesClass надає методи доступу та редагування даних. Також він використовується для зберігання завантажених часових рядів.

Розглянемо параметри, які є атрибутами класу:

- Data_Arrays – список масивів для збереження завантажених часових рядів;
- data_Table – таблиця для зберігання номеру, назви та довжини часових рядів.

Методи класу:

- Add_DataAr(string, double[]) – додає ряд в масив dataAr;
- Add_Elements(int,int) – додає вказану кількість невизначених елементів у кінець вказаного ряду;
- Count_Rows() – повертає кількість завантажених рядів;

- Del_DataAr(int) – видаляє вказаний ряд;
- Del_Element(int,int) – видаляє вказаний елемент із вказаного ряду;
- Edit_Element(int,int,double) – присвоює нове значення елементу ряду;
- Edit_NameRow(int,string) – змінює ім’я ряду;
- Element(int,int) – повертає значення заданого елементу ряду;
- Index_Row(string) – повертає індекс ряду по його імені;
- Lenght_Row(int) – повертає довжину вказаного ряду;
- Name_Row(int) – повертає ім’я ряду по його індексу;
- Replace_DataAr(int ,double[]) – замінює часовий ряд за його індексом;
- Return_Row(int) – повертає масив елементів часового ряду за його індексом;
- Row_Names() – повертає масив імен усіх рядів.

Клас Main_Window утворює головне вікно програми. В ньому створюються об’єкти усіх інших класів.

Методи класу:

- Menu_Item_Click_Load_TimeSeries() – зчитує дані із текстового файлу та зберігає їх;
- Menu_Item_Click_Add_Empty_TimeSeries() – додає порожній ряд;
- Menu_Item_Click_Copy_TimeSeries() – копіює обрані часові ряди;
- Menu_Item_Click_Delete_TimeSeries() – видаляє обрані часові ряди;
- Menu_Item_Click_Show_TimeSeries () – викликає вікно для перегляду і редагування часових рядів;
- Menu_Item_Click_Save_TimeSeries() – зберігає обрані часові ряди у файл;
- Menu_Item_Click_Show_Statistic() – показує статистичні характеристики (математичне сподівання, дисперсію, асиметрію та ексцес) обраних часових рядів;
- Menu_Item_Click_Show_Correlation() – викликає вікно перегляду кореляційної матриці вибраних рядів (за умови, що в них однакова розмірність);

- Menu_Item_Click_Build_Model() – викликає вікно ModelWindow для побудови вибраного типу моделі;
- Menu_Item_Click_Show_CorrelationFunctions() – викликає вікно, де відображається АКФ і ЧАКФ;
- Menu_Item_Click_Build_MovingAverage() – викликає вікно побудови ковзного середнього;
- Menu_Item_Click_Exit() – закриває програму.

Клас Time_Series_Show_Window використовується для перегляду та редагування елементів рядів.

Методи класу:

- Time_Series_Show_Window(int[] indexes) – ініціалізує вікно для роботи з обраними рядами;
- ButtonClick_Save_TimeSeries() – зберігає зміни внесені в обрані часові ряди;
- ButtonClick_Rename_TimeSeries() – перейменовує часовий ряд;
- ButtonClick_Manipulate_TimeSeries() – відкриває вікно для перетворення (логарифмування, нормування і взяття першої різниці) часових рядів;
- ButtonClick_Operations_TimeSeries() – відкриває вікно для лінійних операцій (по елементна сума, різниця, добуток, ділення) над рядами;
- ButtonClick_Show_TimeSeries() – відображає графік обраних часових рядів.

Клас Statistics містить такі методи:

- Math_Averages() – повертає математичні сподівання обраних часових рядів;
- Variances() – повертає дисперсії обраних часових рядів;
- Skewnesses() – повертає коефіцієнти асиметрії обраних часових рядів;
- Kurtosises() – повертає ексцеси обраних часових рядів;
- Kovariations() – повертає коваріаційну матрицю обраних часових рядів;
- Korelations() – повертає кореляційну матрицю обраних часових рядів;

- ACF(int) – повертає автокореляційну функцію обраних часових рядів для заданого ряду;
- PACF(int) – повертає часткову автокореляційну функцію обраних часових рядів для заданого ряду.

Перерахування ModelTypes включає перелік типів моделей і використовується при ініціалізації вікна побудови моделі.

Абстрактний клас AbstractModels реалізує макет моделі (АРКС, АРІКС) та містить наступні атрибути:

- масив a – збереження коефіцієнтів авторегресійної частини;
- масив b –збереження коефіцієнтів частини моделі, пов'язаної з ковзним середнім;
- масив indexes_a – збереження лагів авторегресійної частини моделі, які потрібно включити до моделі;
- масив indexes_b – збереження лагів частини моделі, пов'язаної з ковзним середнім, які потрібно включити до моделі.

Методи класу:

- Abstract_Model() – ініціалізація атрибутів класу;
- X() – повертає матричне представлення правої частини моделі, яке використовується при оцінці коефіцієнтів моделі;
- Y() – повертає матричне представлення лівої частини моделі, яке використовується при оцінці коефіцієнтів моделі;
- init() – повертає ініціалізує коефіцієнти моделі по вектору, отриманого в результаті оцінювання.
- forecast_dynamic() – повертає масив динамічних прогнозів;
- forecast_static() – повертає масив статичних прогнозів;
- ToString() – повертає текстове представлення моделі.

Клас ARMAmodel успадковує клас AbstractModel. Реалізовує всі абстрактні методи базового класу. Крім цього має додатковий конструктор для реалізації моделі АР(p). Модель КС(q) можна отримати, якщо побудувати модель АРКС(0,q).

Клас ARIMA_model успадковує клас AbstractModel. Реалізовує всі абстрактні методи базового класу. Також має додатковий атрибут d для збереження порядку тренду. Крім цього містить два додаткові методи:

- difference() – повертає першу різницю часового ряду;
- sumar() – реалізую операцію обернену взяттю першої різниці.

Інтерфейс AssessmentMethodInterface є макетом для методів оцінки коефіцієнтів моделі. Дозволяє не дублювати код для різних методів оцінювання. Містить один метод teta(), який повертає результат оцінювання. Класи MLS_ і RMLS_ реалізують інтерфейс Assessment_MethodInterface і являють собою метод найменших квадратів (МНК) і РМНК відповідно.

Інтерфейс MovingAverageInterface є макетом для побудови ковзного середнього та дозволяє не дублювати код для різних методів побудови ковзного середнього. Містить один метод ma(), який повертає результат оцінювання.

Методами побудови простого КС, експоненційного КС зі збільшенням та зменшенням вагових коефіцієнтів є класи SimpleMovingAverage, ExponentialMovingAverageDecrease, ExponentialMovingAverageIncrease відповідно.

Реалізація програми є досить гнучкою, даний рівень абстракції дозволяє додавати нові моделі, шляхи обчислення ковзного середнього та оцінки параметрів.

3.3 Функціональні можливості реалізованої СППР

3.3.1 Завантаження даних

Першим способом завантаження даних до програми є: головне меню «Файл»→ «Робота з рядами» → «Завантажити ряд». В діалоговому вікні обираємо текстовий файл з даними. Для успішного виконання команди необхідно, що дані в файлі були числові та розміщені в стовпчик.

Другим способом є введення даних вручну. Головне меню «Файл» → «Робота з рядами» → «Додати порожній ряд». Вказуємо розмірність ряду, далі його назву.

Для роботи з рядами передбачені також додаткові функції, такі як копіювання, видалення та збереження у текстовий файл. У таблиці «Список рядів» можна обрати необхідні ряди та виконувати з ними будь-які операції.

На рисунку 3.3 зображено головне вікно програми після завантаження в неї часових рядів.

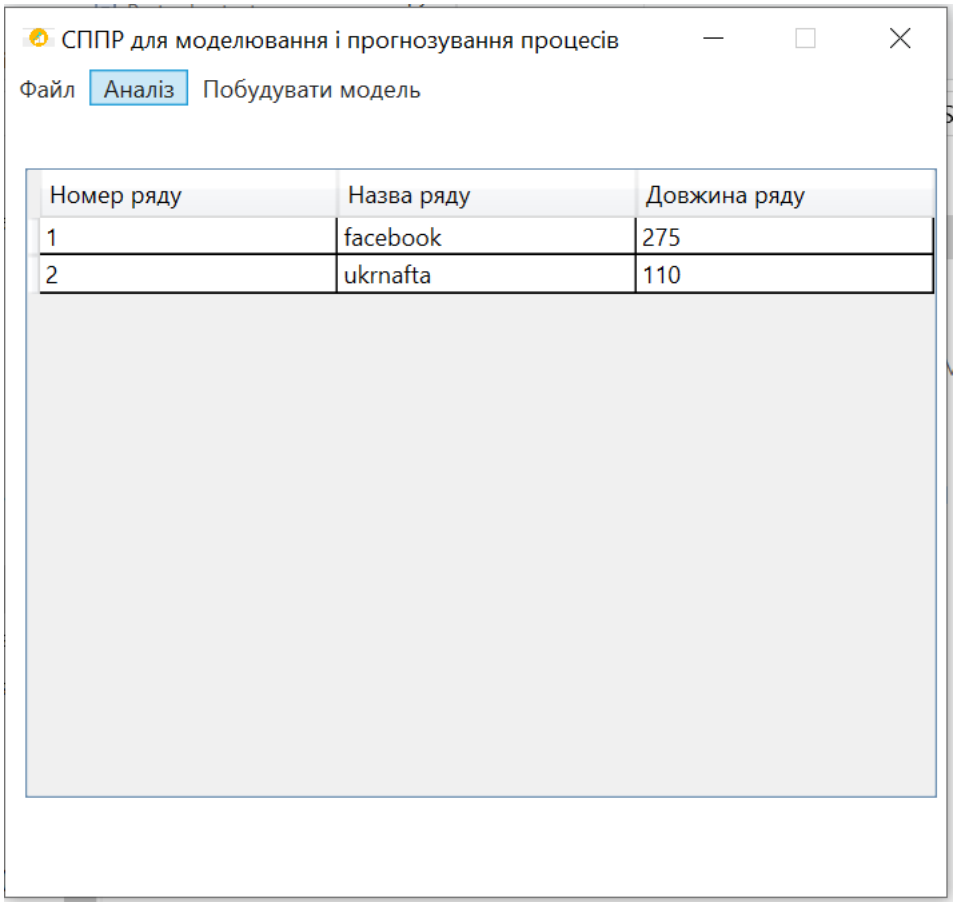


Рисунок 3.3 – Головне вікно програми

3.3.2 Редагування та попередня обробка даних

Щоб відкрити ряд для перегляду та редагування потрібно виконати команду меню «Файл» → «Операції з рядами» → «Редагувати». Можливості вікна для обробки даних (рисунок 3.4):

- додавання та видалення елементів ряду;

- нормування чи логарифмування даних;
- обчислення різниці;
- побудова графіків;
- перейменування ряду;
- збереження змін.

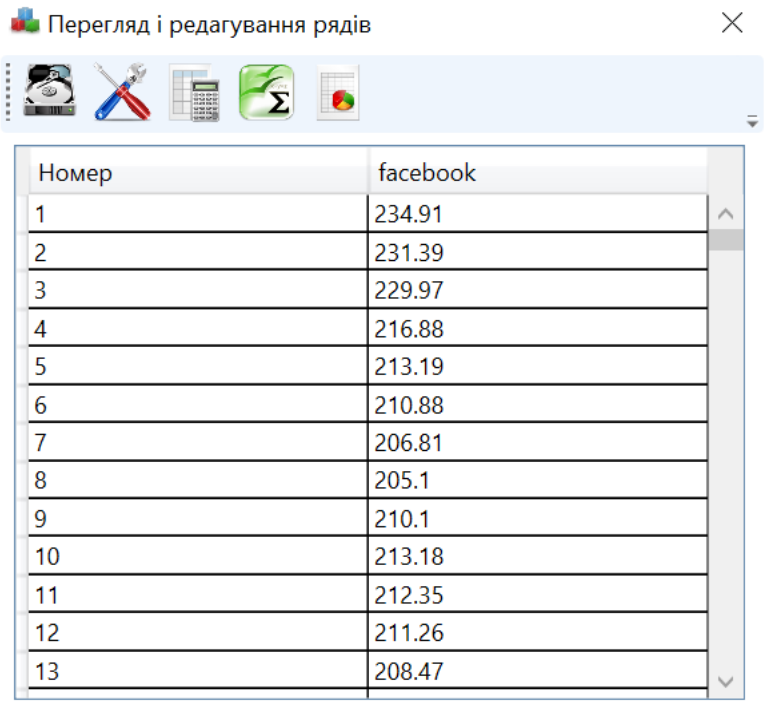


Рисунок 3.4 – Вікно редагування

3.3.3 Графічне представлення даних

Щоб побудувати графік даних потрібно виконати команду меню «Файл» → «Операції з рядами» → «Редагувати». Далі натиснути меню «Графік» вікна редагування даних (рисунок 3.5).

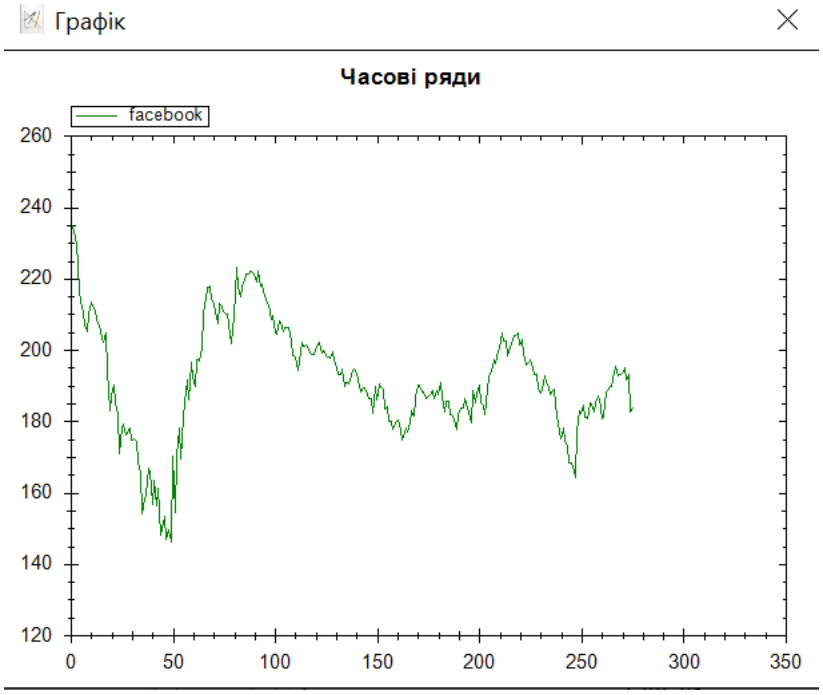


Рисунок 3.5 – Графічне представлення даних

3.3.4 Аналіз даних

Для аналізу даних потрібно обрати потрібний ряд з таблиці та натиснути меню «Аналіз» головного вікна програми.

Розглянемо основні можливості аналізу:

- обчислення описових статистик;
- обчислення коваріаційної та кореляційної матриці;
- обчислення АКФ та ЧАКФ;
- побудова ковзного середнього.

Описові статистики, що розраховуються:

- математичне сподівання;
- дисперсія;
- коефіцієнт асиметрії;
- ексцес.

На рисунку 3.6 наведений приклад обчислення описових статистик ряду.

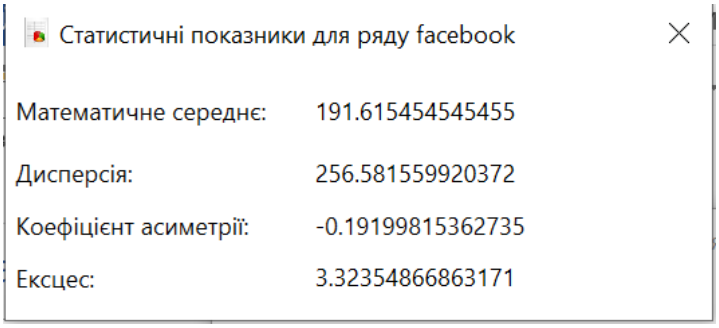


Рисунок 3.6 – Приклад описових статистик ряду

Для побудови АКФ та ЧАКФ ряду необхідно вибрати ряд та виконати команду меню «Аналіз» → «АКФ / ЧАКФ». Приклади побудови АКФ та ЧАКФ наведені на рисунок 3.7 і рисунок 3.8 відповідно.

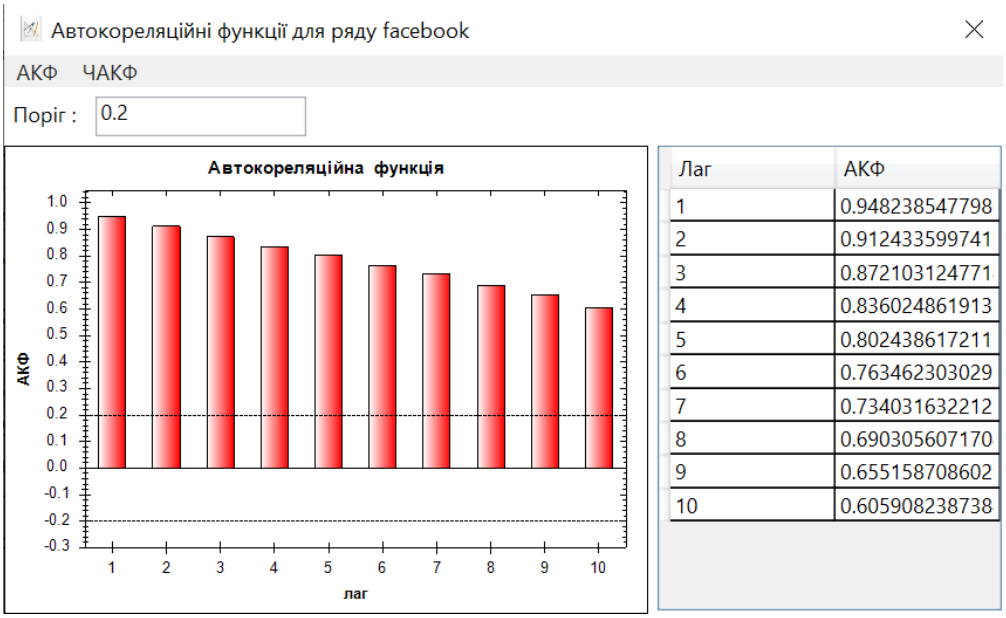


Рисунок 3.7 – Приклад обчислення АКФ

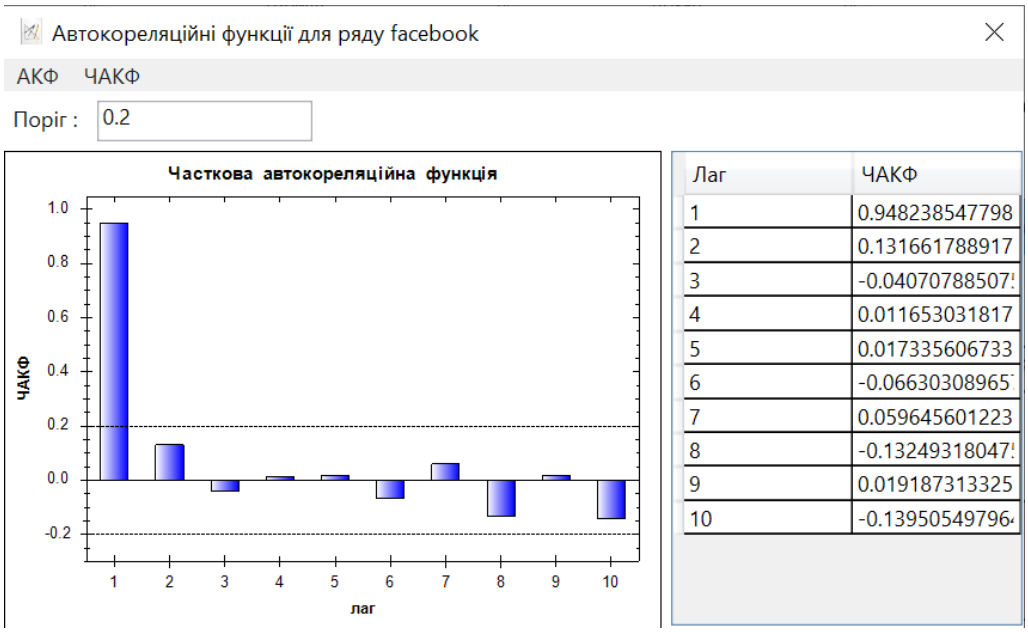


Рисунок 3.8 – Приклад обчислення ЧАКФ

Для побудови КС ряду необхідно вибрати ряд, виконати команду меню «Файл» → «Ковзне середнє» та обрати потрібний тип КС. Приклад побудови простого ковзного середнього наведений на рисунку 3.9.

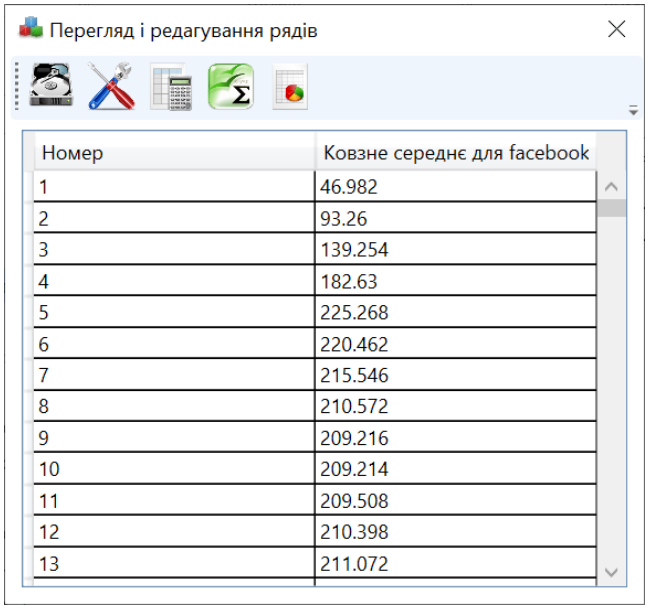


Рисунок 3.9 – Приклад побудови простого КС

3.3.5 Побудова та вибір кращої моделі АРКС

Для виклику вікна побудови моделі АРКС (рисунок 3.10) необхідно натиснути меню головного вікна «Моделі» → «АРКС».

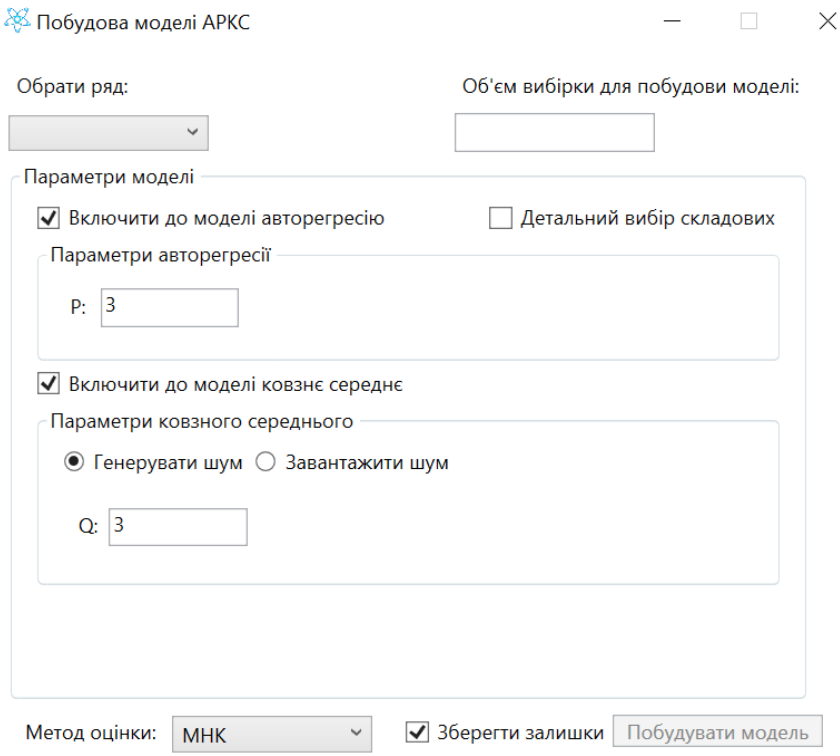


Рисунок 3.10 – Вікно побудови моделі АРКС

Спочатку нам потрібно обрати ряд, для якого будемо будувати модель. Далі вказати, яку частину ряду використовувати для оцінки моделі та параметри які будемо оцінювати. Також можливо вказати, чи завантажувати ковзне середнє, а також метод оцінки параметрів (МНК чи РМНК) і чи зберігати залишки моделі. Після вибору ряду стане активною кнопка «Побудувати модель». Після натискання з'явиться наступне вікно (рисунок 3.11):

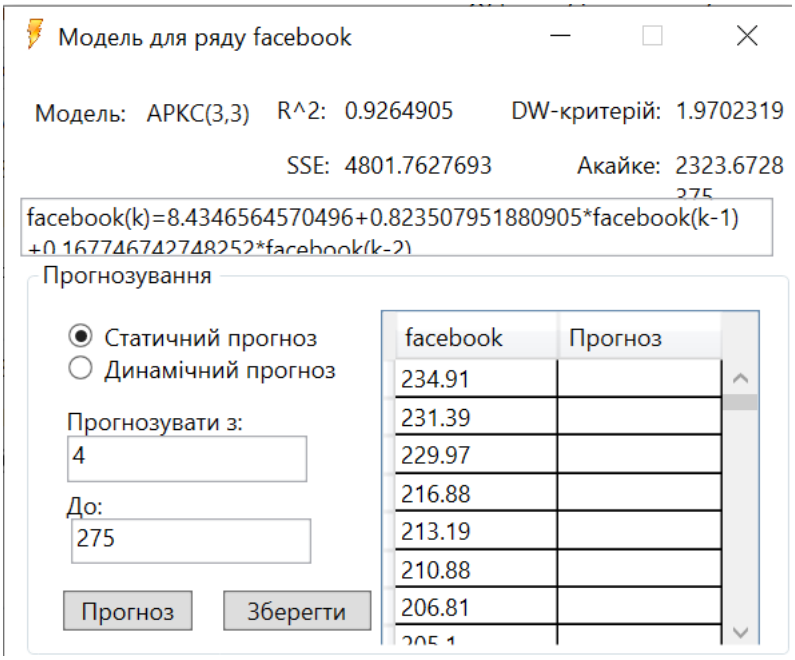


Рисунок 3.11 – Вікно побудованої моделі АРКС

Для порівняння моделей та вибору кращої обчислюються такі характеристики:

- сума квадратів помилок (SSE);
- коефіцієнт детермінації (R^2);
- інформаційний критерій Акайке;
- статистика Дарбіна-Уотсона (DW-критерій).

3.3.6 Прогнозування

Після побудови моделі у вікні (рисунок 3.11) з’являються додаткові кнопки для прогнозу та вибору методу прогнозування.

Спочатку обираємо тип прогнозу: статичний чи динамічний. Далі діапазон для прогнозування: початкова та кінцева точки для прогнозу. В результаті прогнозування відкривається нове вікно з графічним представленням прогнозу і показниками якості прогнозу (рисунок 3.12). Аналогічно передбачена можливість збереження результатів прогнозування натиснувши кнопку «Зберегти».

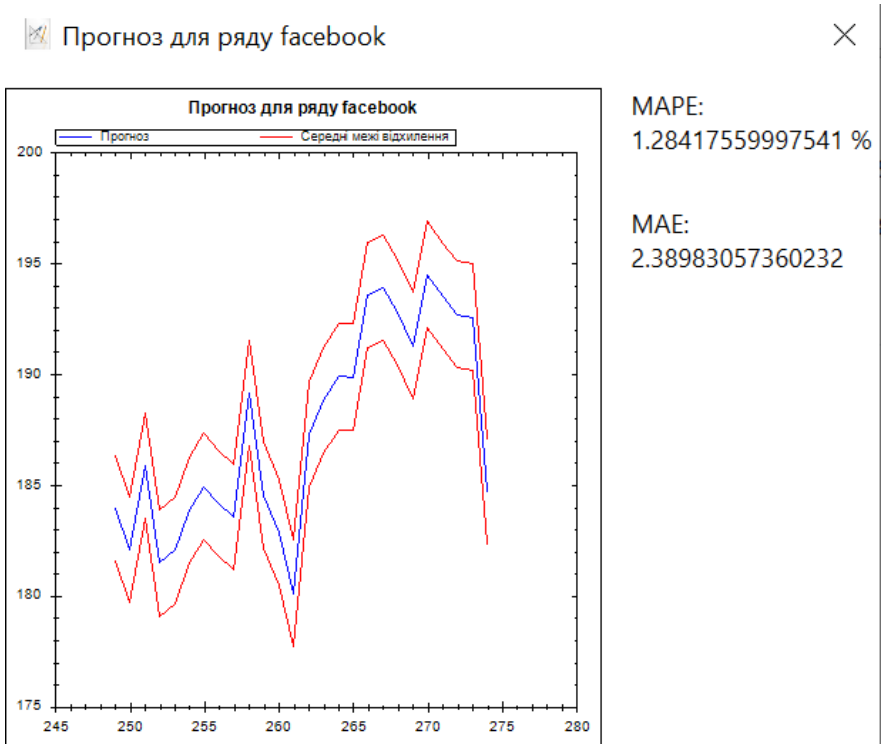


Рисунок 3.12 – Графічне представлення прогнозованих значень і показники якості прогнозу

3.4 Висновки до розділу

В рамках цієї дипломної роботи створена СППР. Система призначена для моделювання та прогнозування нелінійних нестаціонарних процесів в економіці та фінансах.

Програма призначена для надання допомоги в процесі прийняття рішень. Мета створеної системи – підвищення якості та ефективності рішень.

Перевагою створеної СППР є компактність завдяки використанню однієї платформи. З недоліків можна відзначити мале коло питань, що вирішуються за допомогою системи на відміну від аналогів.

РОЗДІЛ 4 МОДЕЛЮВАННЯ І ПРОГНОЗУВАННЯ НЕЛІНІЙНИХ НЕСТАЦІОНАРНИХ ПРОЦЕСІВ

4.1 Дослідження, моделювання та прогнозування цін акцій компанії «Укрнафта»

Протестуємо роботу створеної СППР на прикладі аналізу та моделювання цін акцій. На сайті української біржі отримуємо ціни акцій компанії «Укрнафта» у період з 3 січня 2018 року по 15 червня 2018 року. Аналогічні розрахунки проведемо за допомогою пакету Eviews для порівняння отриманих результатів. На рисунку 4.1 зображений графік цін акцій «Укрнафти» за даних період.

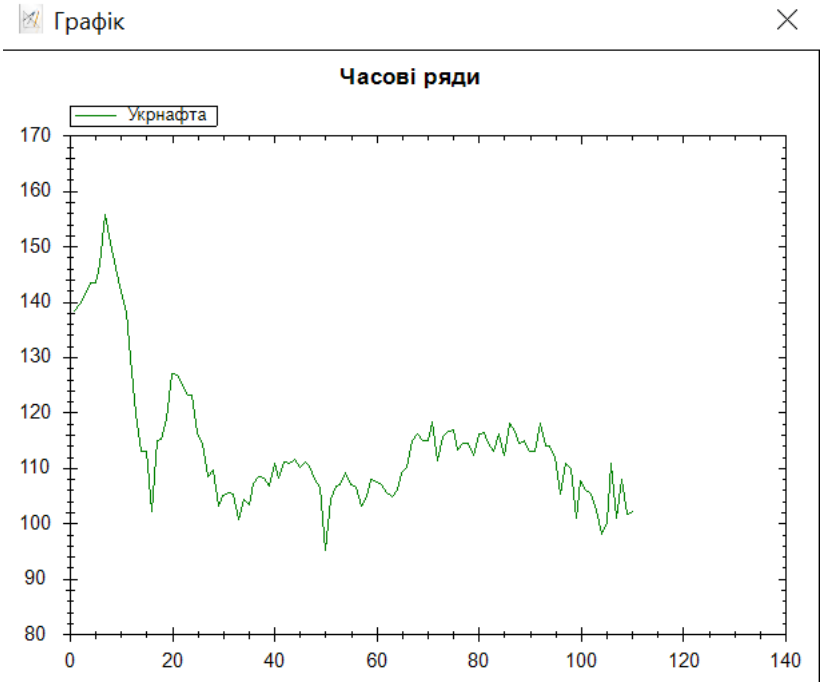
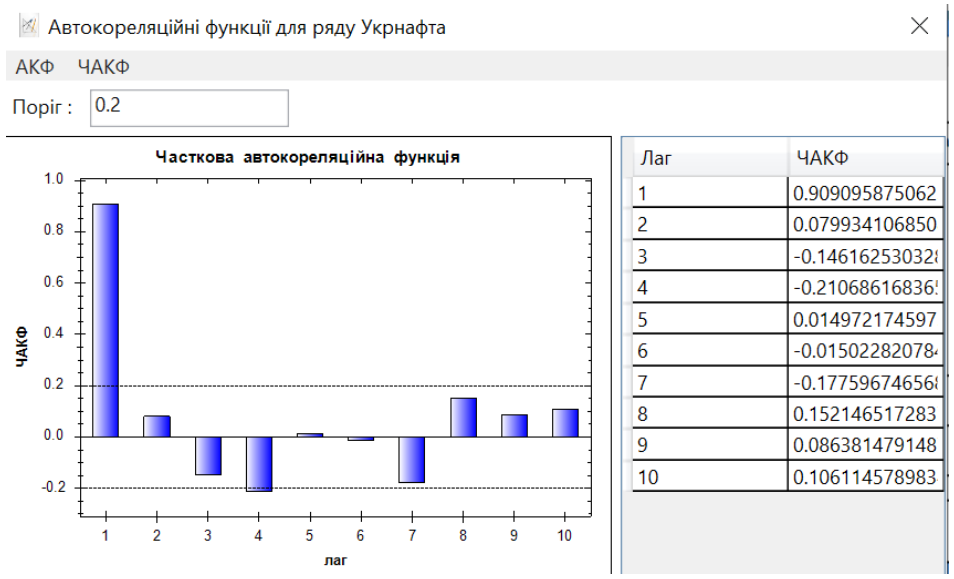
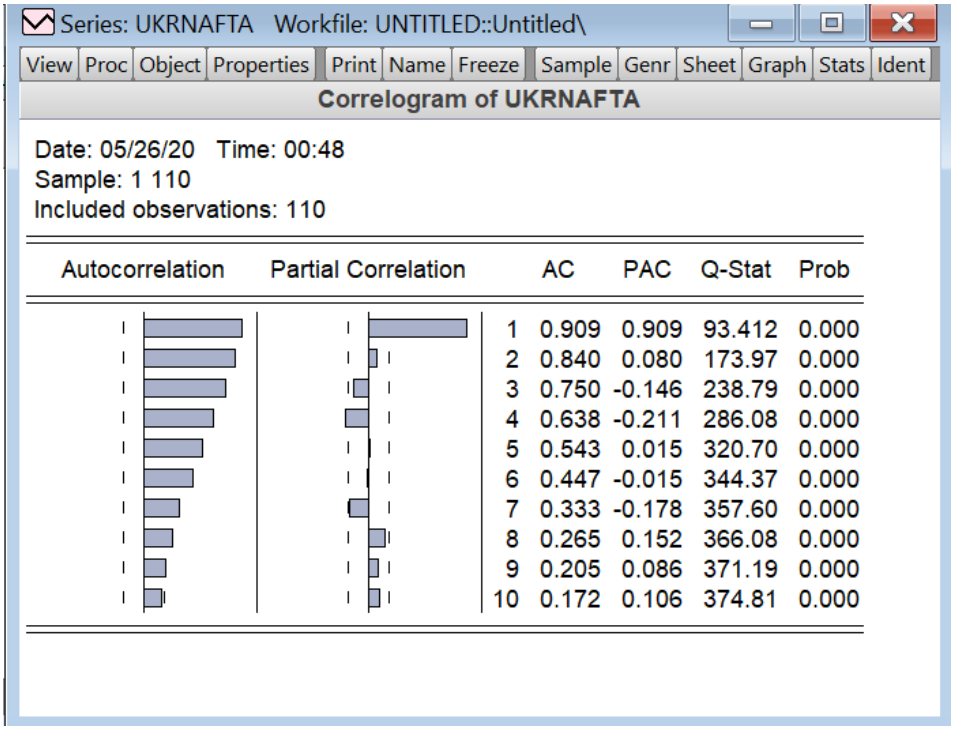


Рисунок 4.1 – Графік цін акцій компанії «Укрнафта»

Побудуємо ЧАКФ для даного процесу за допомогою СППР та пакету Eviews (рисунок 4.2).



а) розроблена СППР



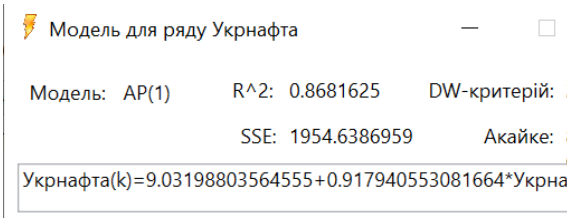
б) Eviews

Рисунок 4.2 – ЧАКФ для цін акцій компанії «Укрнафта»

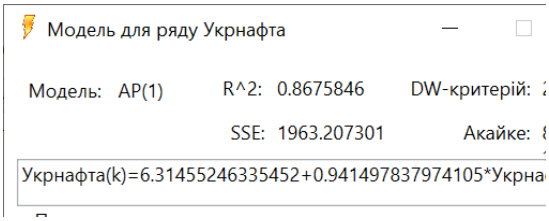
Результати обчислень на СППР (рисунок 4.2, а) і Eviews (Рисунок 4.2, б) майже однакові. Проаналізувавши ЧАКФ будемо будувати модель АР(1) за формулою:

$$y(k) = a_0 + a_1 y(k-1).$$

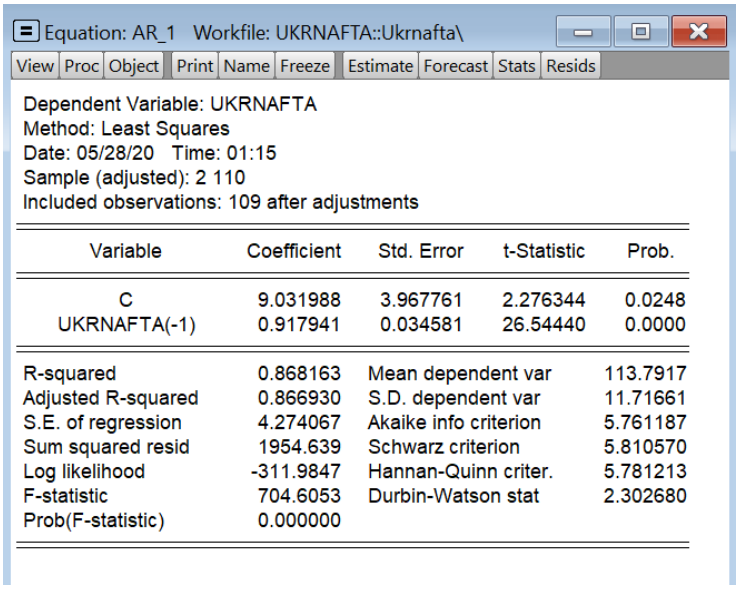
Виконаємо оцінювання коефіцієнтів рівняння методами МНК і РМНК у даній СППР та у Eviews (рисунок 4.3):



а) МНК розроблена СППР



б) РМНК розроблена СППР



в)Eviews

Рисунок 4.3 – Модель AP(1) для цін акцій компанії «Укрнафта»

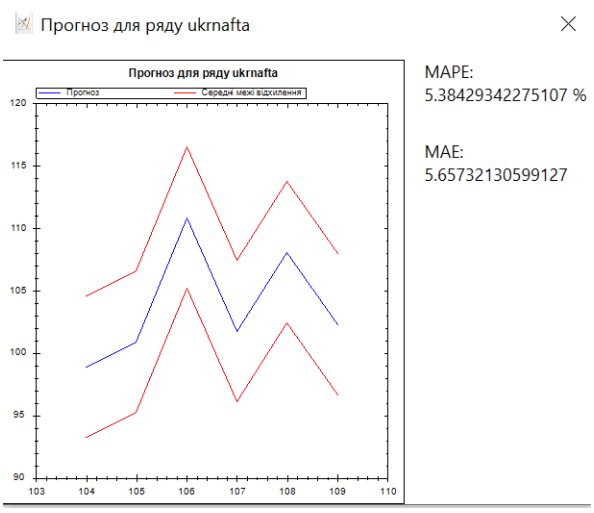
Порівнявши всі три результати робимо висновок, що кращий результат дала розроблена СППР. Найкращий результат дав МНК у розробленій СППР. Результати всіх обчислень занесемо до таблиці 4.1.

Таблиця 4.1 - Оцінки параметрів моделі AP(1) для компанії «Укрнафта»

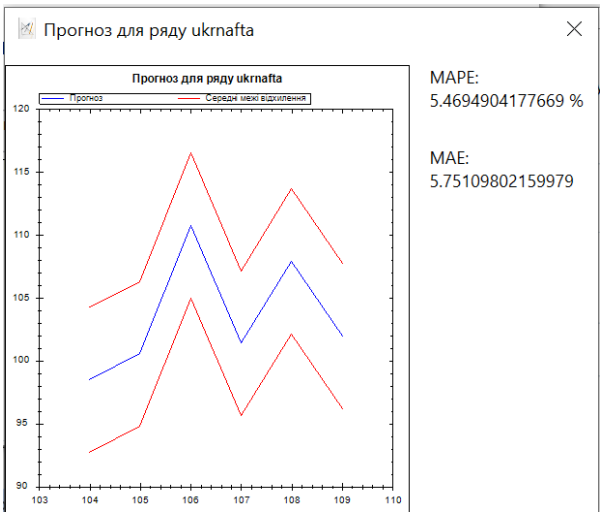
Коефіцієнт	Оцінка в СППР, метод МНК	Оцінка в СППР, метод РМНК	Оцінка в Eviews
a_0	9,031988	6,314552	9,031988
a_1	0,917941	0,941498	0,917941

Коефіцієнт детермінації для всіх трьох випадків є доволі високий.

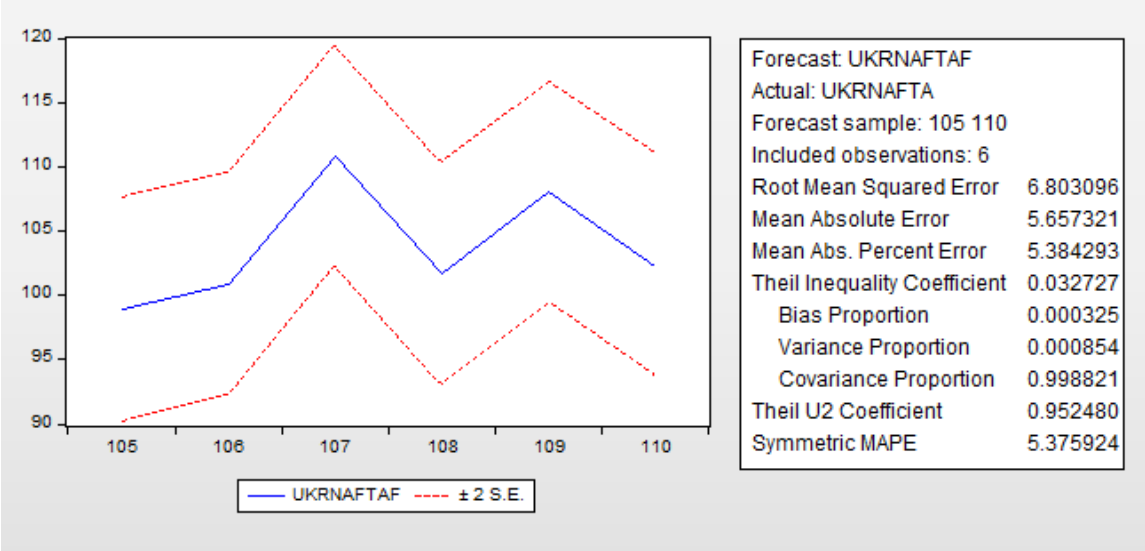
Тепер спробуємо спрогнозувати (статичним методом) п’ять останніх вимірів цін акцій рисунок 4.4:



а) МНК розроблена СППР



б) РМНК розроблена СППР



в)Eviews

Рисунок 4.4 – Прогноз для цін акцій компанії «Укрнафта» на основі моделі АР(1)

Таблиця 4.2 – Результати прогнозування цін акцій компанії «Укрнафта» на основі моделі AP(1)

Вимір	Реальне значення	Розроблена СППР				Eviews	
		МНК		РМНК			
		Прогноз	Похибка	Прогноз	Похибка	Прогноз	Похибка
105	100.1	98.8984	1.6128	98.4872	1.6128	98.8984	1.6128
106	110.9	100.9178	9.9822	100.5585	10.3415	100.9178	9.9822
107	101	110.8316	9.8316	110.7267	9.7267	110.8316	9.8316
108	107.9	101.7439	6.1561	101.4058	6.4942	101.7439	6.1561
109	101.6	108.0778	6.4778	107.9022	6.3022	108.0778	6.4778
110	102	102.2947	0.2947	101.9707	0.0293	102.2947	0.2947
Сер. Похибка			5.7259		5.7511		5.7259
САПП		5.3843%		5.4695%		5.3843%	
САП		5.6573		5.7511		5.6573	

Спробуємо покращити модель побудувавши авторегресійну модель з ковзним середнім порядку 7.

Dependent Variable: RTS1_1
Method: Least Squares
Date: 06/01/20 Time: 21:02
Sample (adjusted): 13 110
Included observations: 98 after adjustments

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	16.40333	7.505493	2.185509	0.0315
RTS1_1(-1)	0.848909	0.067145	12.64294	0.0000
MV(-1)	-1.179843	0.381991	-3.088662	0.0027
MV(-2)	1.396893	0.383874	3.638932	0.0005
MV(-3)	-0.279348	0.362335	-0.770966	0.4428
MV(-4)	-0.140814	0.363026	-0.387891	0.6990
MV(-5)	0.401918	0.369434	1.087929	0.2796
MV(-6)	-1.274060	0.419192	-3.039322	0.0031
MV(-7)	0.442502	0.339089	1.304975	0.1953

R-squared	0.717234	Mean dependent var	110.7326
Adjusted R-squared	0.691817	S.D. dependent var	5.378916
S.E. of regression	2.986062	Akaike info criterion	5.113131
Sum squared resid	793.5746	Schwarz criterion	5.350526
Log likelihood	-241.5434	Hannan-Quinn criter.	5.209152
F-statistic	28.21855	Durbin-Watson stat	1.997005
Prob(F-statistic)	0.000000		

Рисунок 4.5 – Модель АРКС(1,7) для цін акцій компанії «Укрнафта»

Основні характеристики моделі: $R^2 = 0,7172$; $DW = 1,99$; $СКП = 793,57$.

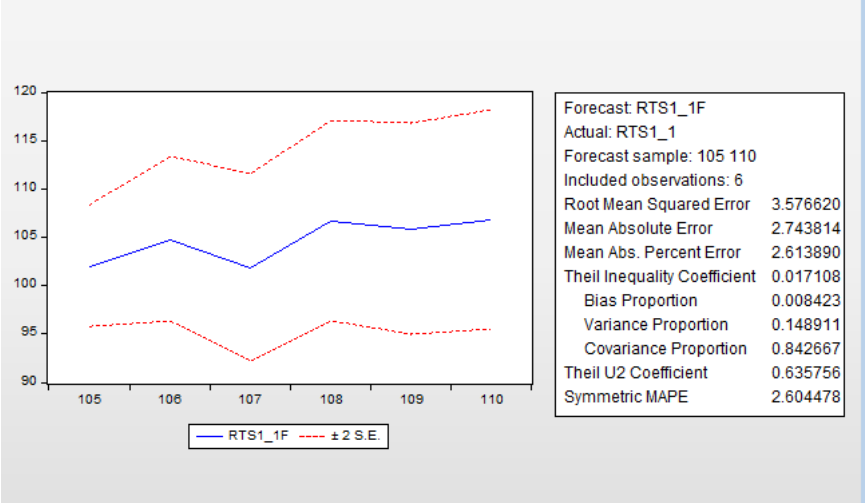


Рисунок 4.6 – Прогноз на основі моделі АРКС(1,7) для цін акцій компанії «Укрнафта»

Остаточний результат моделювання і прогнозування подано у таблиці 4.3 нижче.

Таблиця 4.3 Зведена таблиця результатів моделювання і прогнозування процесу ціноутворення акцій компанії «Укрнафта»

Модель процесу	Характеристики моделі			Характеристики прогнозу			
	R^2	$\sum e^2$	DW	СКП	САП	САПП	Тейла
АР(1)	0,87	1954,64	2,3	6,8	5,65	5,38%	0,03
АРКС(1,7)	0,71	793,57	1,99	3,58	2,74	2,61%	0,02
АР(3)+тренд	0,79	2345,14	2,16	8,31	7,85	7,23%	0,04

4.2 Моделювання і прогнозування цін акцій Facebook

Застосуємо роботу даної СППР для прогнозування ціни акцій компанії «Facebook» на основі даних за 2019 рік. На рисунку 4.5 зображений графік цін акцій компанії за даний період.

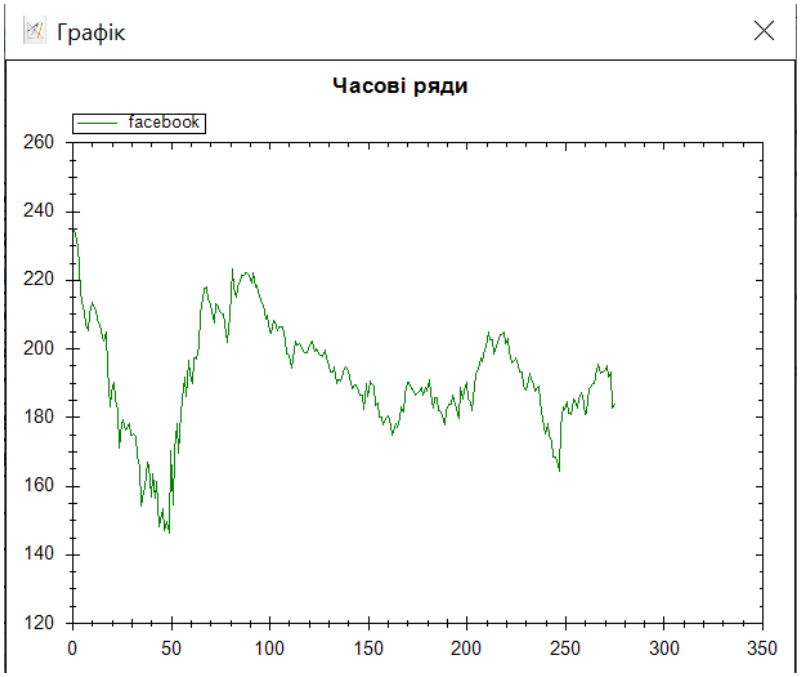


Рисунок 4.7 – Графік цін акцій компанії «Facebook»

Візуальний аналіз свідчить про те, що процес містить складний різнознаковий тренд, який можна описати, наприклад, поліномом третього порядку.

Побудуємо ЧАКФ для даного процесу (рисунок 4.6).

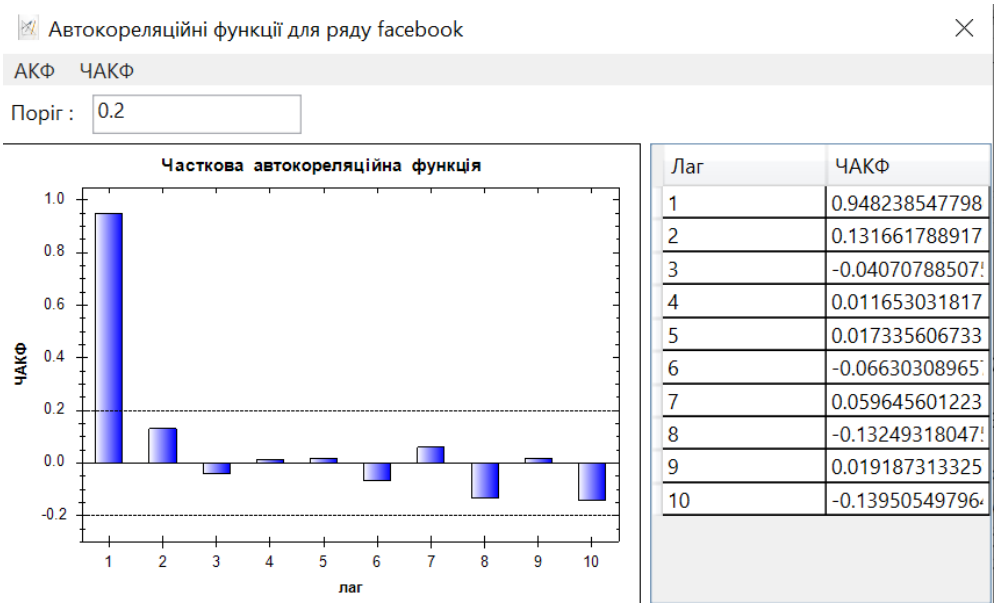
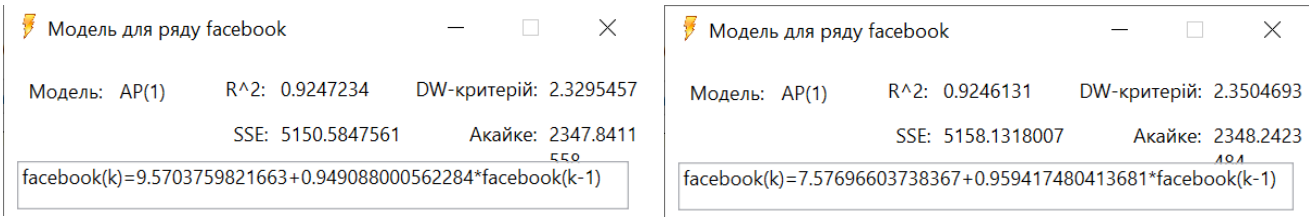


Рисунок 4.8 – ЧАКФ для цін акцій компанії «Facebook»

Проаналізувавши ЧАКФ будемо будувати модель АР(1).

Виконаємо оцінку коефіцієнтів методами МНК і РМНК.



а) МНК б) РМНК

Рисунок 4.9 – Модель АР(1) для цін акцій компанії «Facebook»

Оцінене рівняння процесу методом МНК:

$$x(k) = 9.57 + 0.949 * x(k-1).$$

Основні характеристики моделі: $R^2 = 0,9247;\ DW = 2,33;\ СКП = 5150,58$.

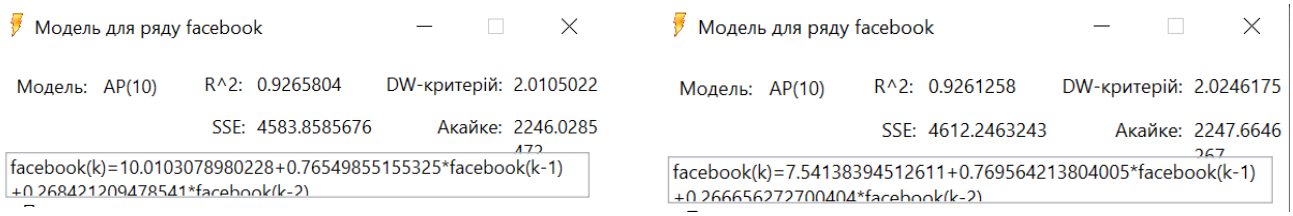
Оцінене рівняння процесу методом РМНК:

$$x(k) = 7.577 + 0.959 * x(k-1).$$

Основні характеристики моделі: $R^2 = 0,9246;\ DW = 2,35;\ СКП = 5158,13$.

Спробуємо покращити модель включенням в авторегресійну модель наступних значень змінної $x(k)$, затриманих в часі: 1, 2, 8, 10.

Виконаємо оцінку коефіцієнтів методами МНК і РМНК.



а) МНК б) РМНК

Рисунок 4.10 – Модель AP(10) для цін акцій компанії «Facebook»

Оцінене рівняння процесу методом МНК:

$$x(k) = 10.017 + 0.763 * x(k-1) + 0.265 * x(k-2) - 0.0222 * x(k-6) - 0.059 * x(k-10).$$

Основні характеристики моделі: $R^2 = 0,9266;\ DW = 2,01;\ СКП = 4583,86$.

Оцінене рівняння процесу методом РМНК:

$$x(k) = 7.541 + 0.769 * x(k-1) + 0.267 * x(k-2) - 0.022 * x(k-6) - 0.054 * x(k-10).$$

Основні характеристики моделі: $R^2 = 0,9261;\ DW = 2,02;\ СКП = 4612,25$.

Побудуємо ЧАКФ для ковзного середнього (рисунок 4.11).

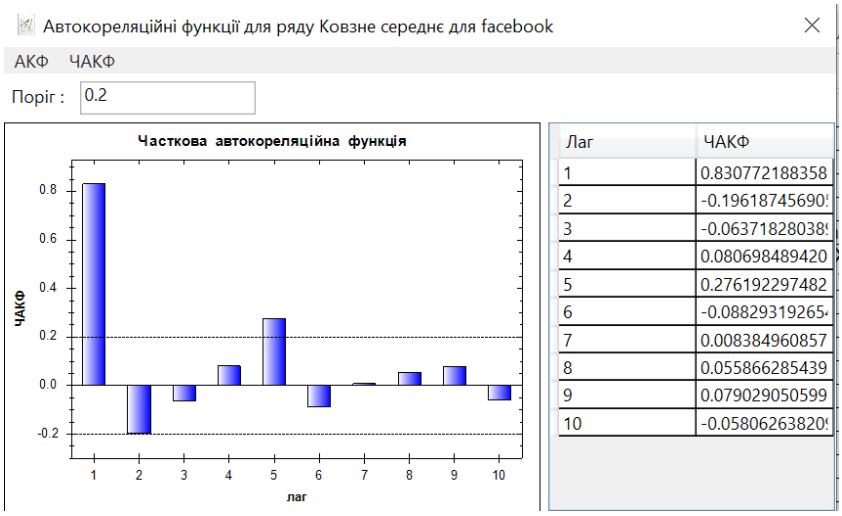
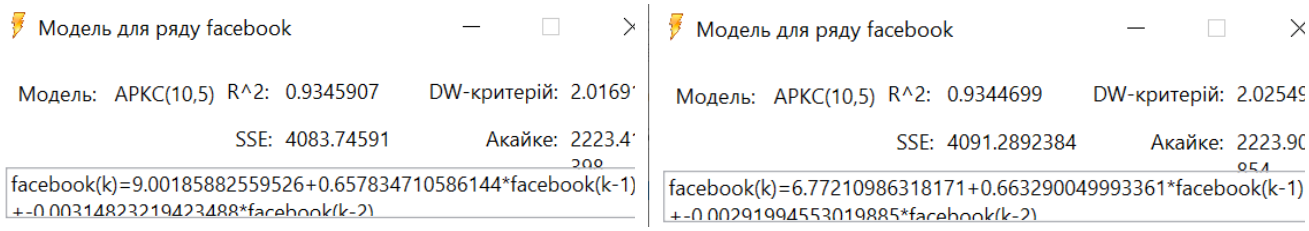


Рисунок 4.11 – ЧАКФ для ковзного середнього цін акцій компанії «Facebook»

Проаналізувавши ЧАКФ, побудуємо модель процесу із врахуванням додаткової шумової складової (ковзного середнього). На рисунку 4.12 зображені основні характеристики моделі АРКС(10,5).



а) МНК б) РМНК

Рисунок 4.12 – Модель АРКС(10, 5) для цін акцій компанії «Facebook»

Оцінене рівняння процесу методом МНК:

$$x(k) = 9.002 + 0.658 * x(k-1) - 0.003 * x(k-2) - 0.370 * x(k-8) - 0.152 * x(k-10) +$$
$$+0.945 * ma(k) - 0.444 * ma(k-1) - 0.319 * ma(k-2) + 0.637 * ma(k-5).$$

Основні характеристики моделі: $R^2 = 0,9346$; $DW = 2,02$; $СКП = 4083,75$.

Оцінене рівняння процесу методом РМНК:

$$x(k) = 6.772 + 0.663 * x(k-1) - 0.003 * x(k-2) - 0.371 * x(k-8) - 0.147 * x(k-10) +$$
$$+0.950 * ma(k) - 0.449 * ma(k-1) - 0.319 * ma(k-2) + 0.641 * ma(k-5).$$

Основні характеристики моделі: $R^2 = 0,9345;\ DW = 2,03;\ СКП = 4091,29$.

Таблиця 4.4 – Характеристики моделей для цін акцій «Facebook»

Модель процесу	Характеристики моделі метод МНК			Характеристики моделі метод РМНК		
	R^2	$\sum e^2$	DW	R^2	$\sum e^2$	DW
АР(1)	0,9247	5150,58	2,33	0,9246	5158,13	2,35
АР(10)	0,9266	4583,86	2,01	0,9261	4612,25	2,02
АРКС(10,5)	0,9346	4083,75	2,02	0,9345	4091,29	2,03

Побудуємо модель АР(3) з трендом. Тренд різнознаковий тому виберемо для його опису поліном третього порядку.

Dependent Variable: FB
Method: Least Squares
Date: 05/29/20 Time: 16:11
Sample(adjusted): 4 250
Included observations: 247 after adjusting endpoints
FB=C(1)+C(2)*FB(-1)+C(3)*FB(-2)+C(4)*FB(-3)+C(5)*K+C(6)*K^2+C(7)*K^3

	Coefficient	Std. Error	t-Statistic	Prob.
C(1)	6.061469	3.529784	1.717235	0.0872
C(2)	0.759652	0.064422	11.79184	0.0000
C(3)	0.229694	0.079701	2.881955	0.0043
C(4)	-0.040313	0.063633	-0.633519	0.5270
C(5)	0.112904	0.042309	2.668549	0.0081
C(6)	-0.000878	0.000382	-2.299546	0.0223
C(7)	1.95E-06	9.85E-07	1.979099	0.0489

R-squared	0.929575	Mean dependent var	191.5285
Adjusted R-squared	0.927814	S.D. dependent var	16.18113
S.E. of regression	4.347444	Akaike info criterion	5.804984
Sum squared resid	4536.064	Schwarz criterion	5.904440
Log likelihood	-709.9155	Durbin-Watson stat	1.979712

Рисунок 4.13 – Модель АР(3) з трендом для цін акцій компанії «Facebook»

Модель:

$$x(k) = 6,06 + 0,759\,x(k-1) + 0,229\,x(k-2) - 0,04\,x(k-3) +$$
$$+ 0,113k - 0,0009k^2 + 1,95E-06\ k^3.$$

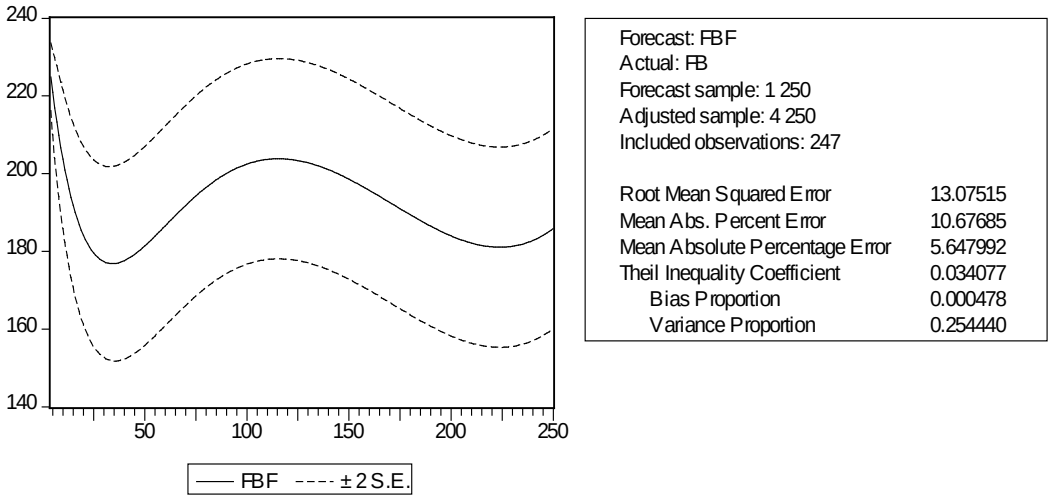


Рисунок 4.14 – Однокроковий прогноз для цін акцій компанії «Facebook» на основі моделі АР(3) з трендом

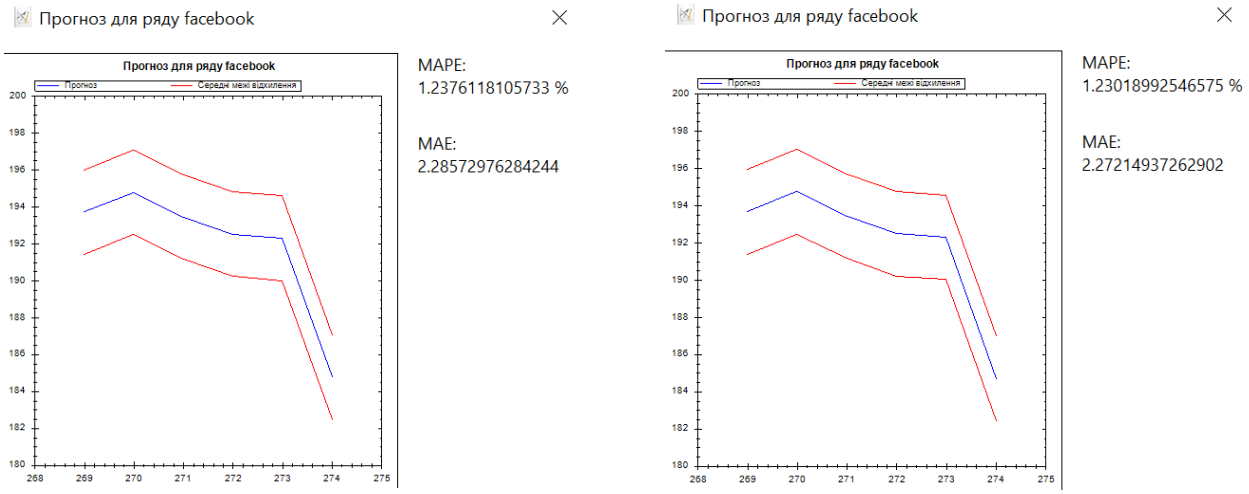
Результат цілком прийнятний, в усякому разі для навчальної вибірки: $САПП = 5,65\%$.

Зведена таблиця отриманих результатів моделювання і прогнозування подана нижче.

Таблиця 4.5 Зведена таблиця результатів для акцій компанії «Facebook»

Модель процесу	Характеристики моделі			Характеристики прогнозу			
	R^2	$\sum e^2$	DW	СКП	САП	САПП	Тейла
АР(1)	0,9247	5150,58	2,33	4,56	3,09	1,65%	0,02
АР(10)	0,9266	4583,86	2,01	4,35	2,59	1,4%	0,01
АРКС(10,5)	0,9346	4083,75	2,02	2,29	2,29	1,24%	0,01
АР(3) +тренд	0,9296	4536,06	1,98	13,08	10,68	5,65%	0,03

Проаналізувавши параметри моделей приходимо до висновку, що найкращою є модель АРКС(10,5). Тепер спробуємо спрогнозувати (статичним методом) п’ять останніх вимірів цін акцій (рисунок 4.15):



а) МНК розроблена СППР б) РМНК

Рисунок 4.15 – Прогноз для цін акцій компанії «Facebook» на основі моделі АРКС(10,5)

Таблиця 4.6 – Результати прогнозування цін акцій компанії «Facebook» на основі моделі АР(10,5)

Вимір	Реальне значення	МНК		РМНК	
		Прогноз	Похибка	Прогноз	Похибка
270	193.4	193.7051	0.3051	193.656	0.255991
271	194.78	194.7685	0.0115	194.7277	0.052306
272	191.49	193.4231	1.9331	193.4125	1.922546
273	193.26	192.4930	0.767	192.4757	0.784261
274	182.58	192.2875	9.7075	192.2762	9.696164
275	183.78	184.7702	0.9902	184.7016	0.921627
Сер. Похибка			2.2857		2.2721
САПП		1.2376%		1.2302%	
САП		2.2857		2.2721	

4.3 Дослідження та моделювання процесу Лоренца

Ряд Лоренца описується рівнянням:

$$\dot{x} = \delta(y - x) \quad \dot{y} = rx - y - xz \quad \dot{z} = xy - bz$$

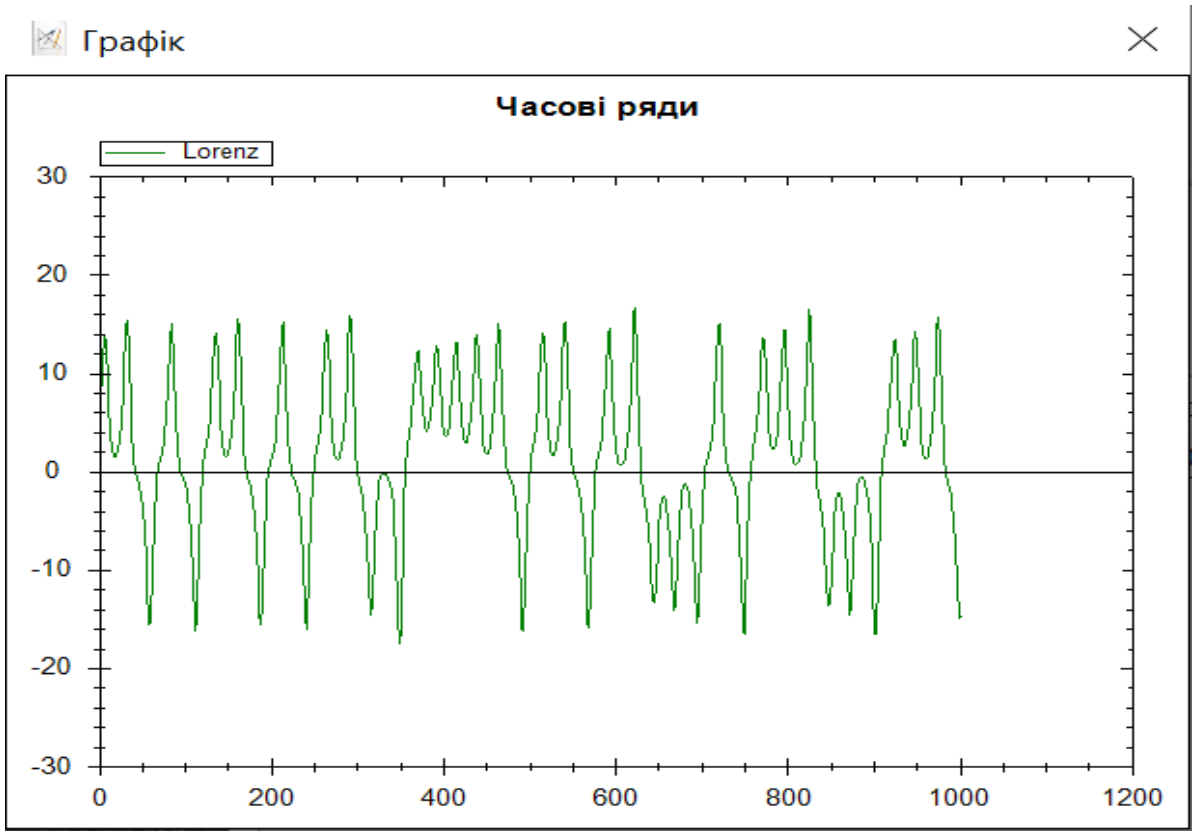


Рисунок 4.16 – Графічне представлення ряду Лоренца

Візуальний аналіз свідчить про те, що процес нелінійний, але спробуємо апроксимувати його динаміку за допомогою лінійної моделі.

Побудуємо ЧАКФ для даного процесу (рисунок 4.17).

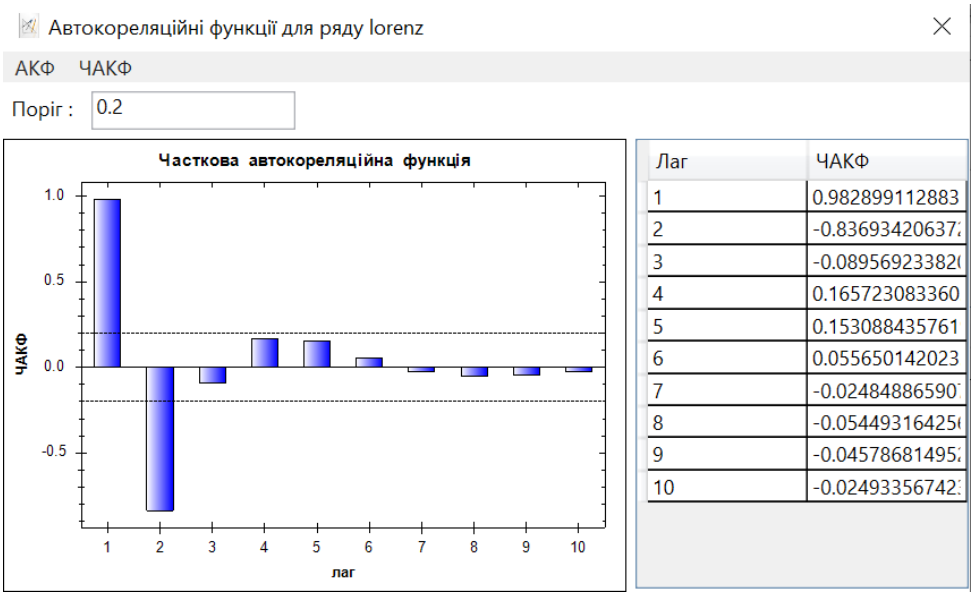


Рисунок 4.17 – ЧАКФ для процесу Лоренца

З отриманої ЧАКФ видно, що в авторегресійну частину моделі необхідно включити наступні значення змінної $x(k)$, затримані в часі: 1 – 6.

Параметри авторегресійної моделі 6-го порядку наведені на рисунку 4.18.

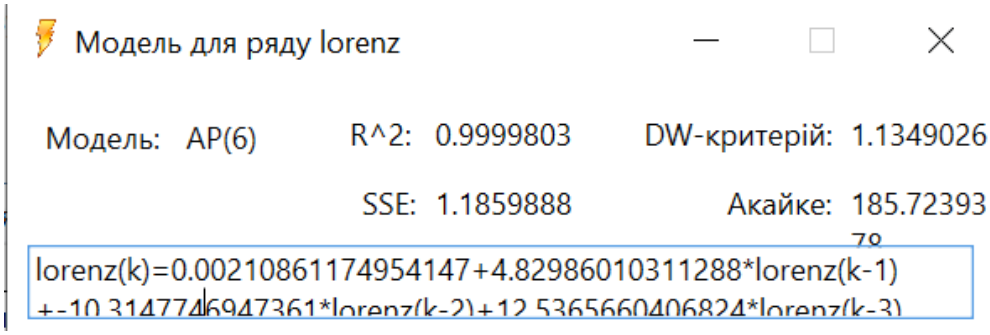


Рисунок 4.18 – Параметри AP(6) процесу Лоренца

Оцінене рівняння процесу:

$$x(k) = 0{,}002 + 4{,}83x(k-1) + 10{,}31x(k-2) + 12{,}54x(k-3) - 9{,}18x(k-4) + \\ + 3{,}84x(k-5) - 0.17x(k-6).$$

Основні статистичні характеристики моделі:

$$R^2 = 0,9999803;\ \ DW = 1,135;\ \ СКП = 1,186.$$

Характеристики однокрокового прогнозу:

$$СКП = 7,98;\quad САП = 6,29;\ САПП = 144,36;\ U = 0,78.$$

Таким чином, за допомогою даної моделі можна отримати прогноз середньої якості, а коефіцієнт Тейла (ідеальне значення = 0) свідчить про існування можливостей покращення прогнозу. Тому побудуємо модель вищого порядку. На рисунку 4.19 зображені характеристики авторегресії 8 порядку.

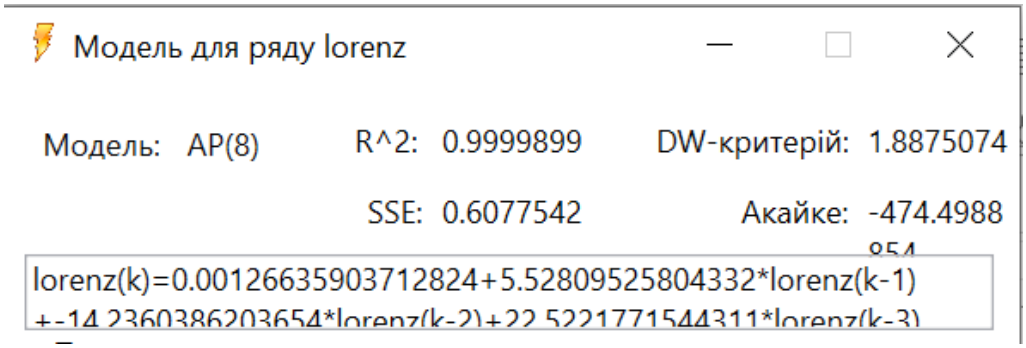


Рисунок 4.19 – Параметри AP(8) процесу Лоренца

Отримана модель 8-го порядку має вигляд:

$$x(k) = 0,001 + 5,53x(k-1) - 14,24x(k-2) + 22,52x(k-3) - 24,13x(k-4) +$$
$$+ 18,03x(k-5) - 9,19x(k-6) + 2,92x(k-7) - 0,44x(k-8).$$

Основні статистичні характеристики моделі:

$$R^2 = 0,9999899;\ \ DW = 1,8875;\ \ СКП = 0,608$$

Характеристики однокрокового прогнозу:

$$СКП = 7,8; \quad САП = 6,13; \; САПП = 136,95;$$

Помітно, що при підвищенні порядку авторегресії спостерігається незначне покращення характеристик прогнозу.

Таблиця 4.7 – Характеристики моделей процесу Лоренца

Модель процесу	Характеристики моделі			Характеристики прогнозу		
	R^2	$\sum e^2$	DW	СКП	САП	САПП
АР6	0,99998	1,186	1,135	7,98	6,29	144,36
АР8	0,99999	0,608	1,8875	7,8	6,13	136,95

Проаналізувавши результати моделювання бачимо, що процес, який описується рядом Лоренца, дав хорошу якість прогнозу.

4.4 Комбінування оцінок прогнозів

4.4.1 Усереднення прогнозів

Найпростішим способом комбінування методів оцінок прогнозів є середнє значення. Формула для обчислення середнього прогнозу:

$$\hat{y}_c(k) = \frac{\hat{y}_1(k) + \hat{y}_2(k)}{2},$$

де $\hat{y}_c(k)$ – комбінований прогноз;

$\hat{y}_1(k)$, $\hat{y}_2(k)$ – прогнози, отримані різними методами.

Комбінований прогноз буде незміщеним при умові незміщеності окремих прогнозів. За формулою 4.2 визначаємо похибку комбінованого прогнозу:

$$e_c(k) = y(k) - \hat{y}_c(k) = y(k) - \frac{\hat{y}_1(k) + \hat{y}_2(k)}{2} = \frac{e_1(k) + e_2(k)}{2},$$

де $y(k)$ – фактичне значення прогнозованої змінної.

Дисперсія похибки комбінованого прогнозу:

$$\text{var}\left[\frac{e_1(k) + e_2(k)}{2}\right] = E\left[\frac{e_1(k) + e_2(k)}{2}\right]^2 = \frac{1}{4}E\left[e_1^2(k) + 2e_1(k)e_2(k) + e_2^2(k)\right] =$$

$$= \frac{1}{4}\left\{E\left[e_1^2(k)\right] + 2E\left[e_1(k)e_2(k)\right] + E\left[e_2^2(k)\right]\right\} =$$

$$= \frac{1}{4}\left[\sigma_1^2 + 2\frac{E[e_1(k)e_2(k)]}{\sigma_1\sigma_2}\sigma_1\sigma_2 + \sigma_2^2\right] = \frac{\sigma_1^2 + 2\rho\sigma_1\sigma_2 + \sigma_2^2}{4}.$$

Запишемо формулу для обчислення дисперсії комбінованого прогнозу:

$$\sigma_c^2 = \frac{\sigma_1^2 + \sigma_2^2 + 2\rho\sigma_1\sigma_2}{4},$$

де ρ – коефіцієнт кореляції між похибками прогнозу. За умові незалежності похибок між двома моделями $\rho = 0$. В даній ситуації формула для обчислення спрощується:

$$\sigma_c^2 = \frac{\sigma_1^2 + \sigma_2^2}{4}.$$

Робимо висновок, що за умови незалежності дисперсій похибок обох моделей дисперсія комбінованої похибки буде значно меншою будь-якої з двох дисперсій. Наприклад, нехай $\sigma_1^2 = \sigma_2^2 = 100$:

$$\sigma_c^2 = \frac{100+100}{4} = 50.$$

Якщо кореляція між похибками прогнозування існує, дисперсія похибки комбінованого прогнозу все одно буде меншою ніж дисперсія кожного методу окремо. Наприклад, нехай $\sigma_1^2 = \sigma_2^2 = 100$ і $\rho = 0.8$:

$$\sigma_c^2 = \frac{\sigma_1^2 + \sigma_2^2 + 2\rho\sigma_1\sigma_2}{4} = \frac{100+100+2\cdot 0.8\cdot 10\cdot 10}{4} = 90.$$

Після усереднення оцінок, отриманих за двома методами, спостерігається зменшення дисперсії похибки прогнозу.

Результат буде гіршим у випадку, коли дисперсії похибок моделей значно відрізняються. Наприклад, нехай $\sigma_1^2 = 100,\ \sigma_2^2 = 16$ і $\rho = 0.8$:

$$\sigma_c^2 = \frac{\sigma_1^2 + \sigma_2^2 + 2\rho\sigma_1\sigma_2}{4} = \frac{100+16+2\cdot 0.8\cdot 10\cdot 4}{4} = 45.$$

Усереднення похибок доцільно робити у випадку, коли дисперсії похибок прогнозування не сильно відрізняються.

4.4.2 Зважене усереднення прогнозів

За умови відсутності інформації щодо характеристик індивідуальних прогнозів, окремим прогнозам присвоюються різні вагові коефіцієнти на основі суб'єктивних суджень:

$$\hat{y}_c(k) = w_1\,\hat{y}_1(k) + w_2\,\hat{y}_2(k),$$

де w_1, w_2 – вагові коефіцієнти. Зазначимо, що більші значення вагових коефіцієнтів потрібно присвоювати тим індивідуальним прогнозам, які мають меншу дисперсію похибок. При цьому для коректності обчислень необхідно, щоб виконувалась умова: $w_1 + w_2 = 1$.

4.4.3 Вибір вагових коефіцієнтів за допомогою похибок прогнозів

Як правило, похибки прогнозів для конкретних моделей і процесів відомі, або їх можна визначити. Це дає можливість об'єктивно підійти до розв'язку задачі вибору вагових коефіцієнтів. За основу для визначення вагових коефіцієнтів приймаємо квадрати похибок прогнозів, оскільки моделі з меншими сумами квадратів дають якісніші прогнози. Позначимо суму квадратів похибок прогнозування через

$$sse = \sum_{k=1}^{N} e^2(k).$$

Записуємо вирази для вагових коефіцієнтів окремих прогнозів:

$$w_1 = \frac{1/sse_1}{1/sse_1 + 1/sse_2},$$

$$w_2 = \frac{1/sse_2}{1/sse_1 + 1/sse_2},$$

де sse_1, sse_2 – суми квадратів похибок для кожного методу.

Наприклад, нехай $sse_1 = 100$, $sse_2 = 16$:

$$w_1 = \frac{1/100}{1/100 + 1/16} = \frac{0.01}{0.01 + 0.0625} = 0.1379,$$

$$w_2 = \frac{1/16}{1/100 + 1/16} = \frac{0.0625}{0.01 + 0.0625} = 0.8621.$$

Таким чином, ми (об'єктивно) присвоїли більший ваговий коефіцієнт точнішому методу прогнозування. При цьому необхідна для коректності застосування методу умова $\sum w_i = 1$ виконується.

4.4.4 Застосування методу усереднення прогнозів у макроекономічних процесах

У таблиці 4.8 наведено фрагмент обчислених значень окремих прогнозів, виконаних на один крок різними методами прогнозування.

Таблиця 4.8 – Числові значення прогнозів

Числові дані	Методи прогнозування				
Y	Y експ	Y експ2	Y МГУА	Y НМГУА	Y Калман
y_i	$\hat{y}_1(k+1)$	$\hat{y}_2(k+1)$	$\hat{y}_3(k+1)$	$\hat{y}_4(k+1)$	$\hat{y}_5(k+1)$
7,6780	7,5222	7,4413	7,6184	7,5555	7,5651
RSME	0,215812	0,379131	0,115582	0,147774	0,400863

Нижче наведені результати обчислених значень ваг і RSME (комбінування результатів різних методів з вагами обернено пропорційними їх дисперсії), розглянуті при комбінуванні у випадках п'яти, чотирьох і трьох окремо взятих методів прогнозування.

Випадок 1. Комбінуємо п'ять окремо взятих методів прогнозування. Для цього випадку наведені обчислені значення ваг для обраних п'яти окремо взятих методів прогнозування:

Таблиця 4.9 – Числові значення ваг для різних методів

Y експ	Y експ2	Y МГУА	Y НМГУА	Y Калман
0.0636182	0.249936	0.194041	0.390993	0.101413

У таблиці 4.10 представлені значення Y і їх оцінки, обчислені значення прогнозу на один крок, виконані різними методами прогнозування для п'яти окремих прогнозів.

Таблиця 4.10 – Значення та оцінки прогнозу на один крок

Y експ	Y експ2	Y МГУА	Y НМГУА	Y Калман	Y combo
0.0160133	0.00807901	0.00916907	0.00645933	0.0126831	0.00507773

Значення комбінованого прогнозу 0.00507773.

Випадок 2. Комбінують чотири окремо взятих методи прогнозування.

Представлені значення Y і їх оцінки, обчислених значень прогнозу на один крок, виконаних різними методами прогнозування для чотирьох окремо взятих методів.

Таблиця 4.11 – Значення та оцінки прогнозу на один крок

Y эксп	Y эксп2	Y МГУА	Y НМГУА	Y combo
0.0160133	0.00807901	0.00916907	0.00645933	0.00507872

Випадок 3. Поєднано три окремо взяті методи прогнозування.

З урахуванням обчислених значень ваг окремих прогнозів наведені значення Y і їх оцінки, виконаних на один крок різними методами прогнозування. Аналогічно, обчислені значення RSME отримані в наступному вигляді (таблиці 4.12):

Таблиця 4.12 – Значення та оцінки прогнозу на один крок

Y эксп	Y эксп2	Y МГУА	Y combo
0.0160133	0.00807901	0.00916907	0.00567565

Неважко помітити, що значення обчисленого комбінованого прогнозу погіршилося.

Проведені експериментальні дослідження дозволяють зробити наступні висновки: зі збільшенням кількості методів прогнозування якість комбінованого

прогнозу поліпшується і наближається до точного значення. Але у подальших дослідженнях необхідно встановити скільки методів доцільно брати.

4.5 Висновки до розділу

Розроблену СППР протестовано на ряді, що описує ціну акцій компанії «Укрнафта» і отримані результати порівняно із аналогічними обчисленнями у системі Eviews. За результатами порівняння дійшли до висновку, що реалізований програмний продукт за рівнем якості обчислень не поступається уже існуючому аналогу.

Виконано аналіз та моделювання процесів ціноутворення акцій компаній «Укрнафта», «Facebook» та процесу Лоренца. Використано методи ідентифікації структури моделі (на основі автокореляційних функцій). Побудовано моделі процесів та прогнозування на декілька кроків.

Порівняно моделі автокореляції та автокореляції з ковзним середнім при обчисленні методами МНК та РМНК. Для прогнозування обрано найкращі за всіма параметрами моделі.

Таким чином, деякі нелінійні процеси можна з прийнятним ступенем адекватності описати лінійними моделями АРКС відносно високих порядків та використати ці моделі для обчислення прогнозів.

Розглянуто метод комбінування оцінок як один із варіантів покращення якості моделювання та прогнозування. Встановлено, що якість прогнозів покращується із збільшенням методів прогнозування, але потрібно встановити порогове значення для кількості методів, що використовується.

РОЗДІЛ 5 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

5.1 Постановка завдання проектування

Розробити та оцінити програмний продукт, призначений для моделювання і прогнозування нелінійних нестаціонарних процесів в економіці та фінансах. Програмний продукт було розроблено на мові програмування C#.

5.2 Обґрунтування функцій програмного продукту

Виходячи з постановки завдання виділимо наступні основні функції програмного продукту:

- F_1 – мова програмування;
- F_2 – завантаження вхідних даних;
- F_3 – інтерфейс користувача.

Розглянемо різні варіанти реалізацій кожної функції.

Функція F_1:

а) мова програмування C#;

б) мова програмування JavaScript;

Функція F_2:

а) введення даних вручну (з файлу з певним форматом даних);

б) написання нового модулю розпізнавання даних.

Функція F_3:

а) інтерфейс користувача, створений за технологією WPF;

б) інтерфейс користувача, створений за технологією Windows Forms.

На рисунку 5.1 зображена морфологічна карта, яка показує комбінації всіх варіантів реалізації, які формують повну множину варіантів програмного продукту.

На основі морфологічної карти сформована позитивно-негативна матриця варіантів основних функцій (таблиця 5.1).

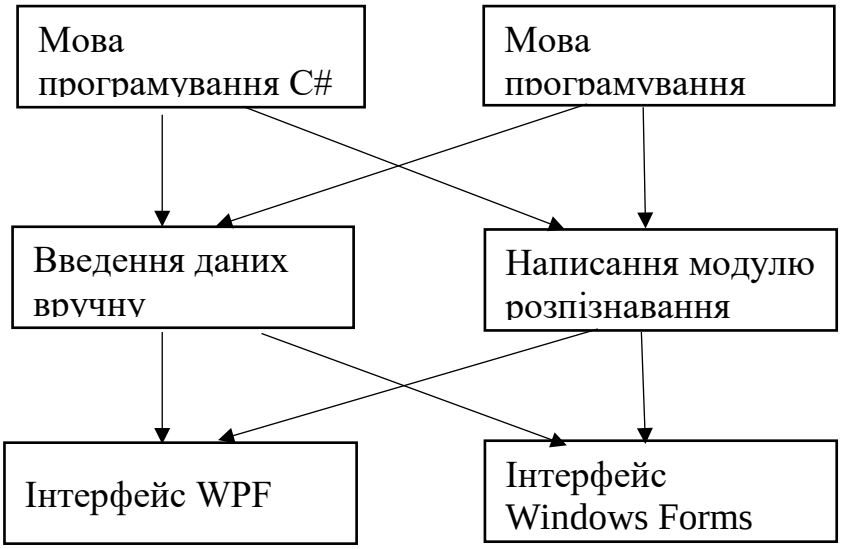


Рисунок 5.1 – Морфологічна карта

Таблиця 5.1 – Позитивно-негативна матриці

Функції	Варіанти	Переваги	Недоліки
F1	А	Швидке написання коду	Не кросплатформний
	Б	Код кросплатформний	Багато коду
F2	А	Простий у використанні	Мала точність обчислень
	Б	Зручне, розумне введення	Багато затраченого часу
F3	А	Стабільний	Додаткова інсталяція
	Б	Легкий у використанні	Не кросплатформний

Проаналізувавши таблицю робимо висновок про відкидання деяких реалізацій, адже вони не відповідають поставленим завданням. Визначаємо складові, які необхідно виключити:

Функція F1:

Оскільки для розрахунків використовуються великі обсяги вхідних даних, то час виконання програмного коду є дуже важливим, тому варіант б) відкидаємо.

Функція F2:

Оскільки програма націлена на роботу з різними часовими рядами, то перший варіант є більш зручним в даному випадку, а варіант б) має бути відкинутий.

Функція F3:

Для роботи з числовими даними інтерфейс не відіграє важливу роль, тому будемо розглядати обидва варіанти.

Формуємо два різні варіанти реалізації програмного продукту:

а) F1а – F2а – F3а;

б) F1а – F2а – F3б.

5.3 Обґрунтування системи параметрів ПП

Визначаємо параметри для характеристики ПП на основі побудованих функцій.

Створений програмний продукт характеризують такі параметри:

- X1 – швидкодія мови програмування;
- X2 – об’єм оперативної пам’яті, необхідний для роботи програми;
- X3 – час, затрачений на обробку даних;
- X4 – кількість рядків програмного коду.

Діапазон значень параметрів показано у таблиці 5.2.

Таблиця 5.2 – Діапазон значень параметрів ПП

Назва параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірше	середнє	краще
F1	X1	Оп/мс	19000	11000	2000
F2	X2	Мб	24	20	16
F3	X3	мс	600	400	100
F3	X4	кількість рядків коду	4000	3000	2000

Визначимо функцію F3 як суму X1+X2.

Побудуємо графічні характеристики параметрів за даними таблиці 5.2 (рисунок 5.2).

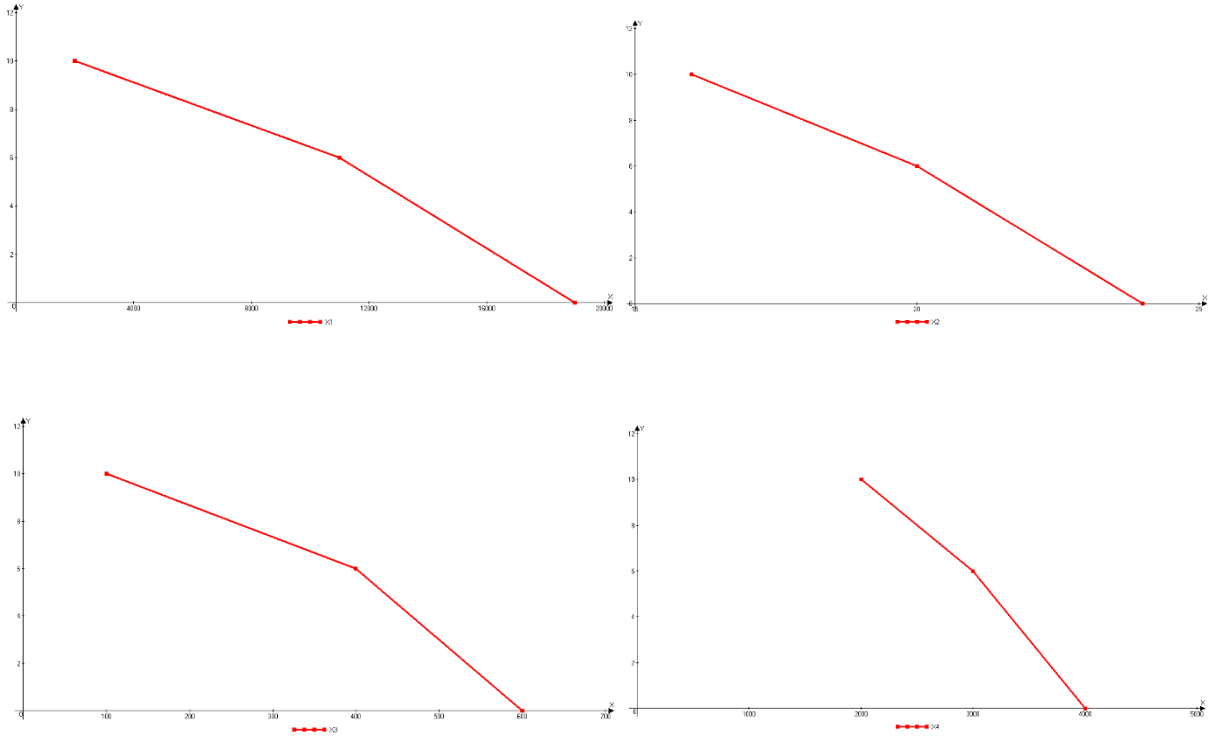


Рисунок 5.2 – Графічні характеристики параметрів

5.4 Аналіз експертного оцінювання параметрів

Для проведення експертного ранжування складаємо комісію із 7 людей. У таблиці 5.3 наводимо результати експертного ранжування.

Таблиця 5.3 – Результати ранжування параметрі

Позначення параметра	Одиниця виміру	Ранг за оцінкою експерта							R_i	Δ_i	Δ_i^2
		1	2	3	4	5	6	7			
X1	Оп/мс	4	3	3	3	2	3	3	21	3,5	12,25
X2	Мб	1	2	2	2	3	2	2	14	-3,5	12,25
X3	Мс	2	1	1	1	1	1	1	8	-9,5	90,25
X4	шт	3	4	4	4	4	4	4	27	9,5	90,25
	Разом	10	10	10	10	10	10	10	70	0	205

За найбільший ранг беремо 4, за найменший – 1.

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 205}{7^2(4^3 - 4)} = 0,84 > W_k = 0,67 .$$

Коефіцієнт узгодженості більше нормативного значення 0,67, тому результати вважаємо достовірними.

Проведемо попарне порівняння параметрів та результати занесемо у таблицю 5.4.

Таблиця 5.4 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	<	<	<	<	>	<	<	<	0,5
X1 і X3	<	<	<	<	<	<	<	<	0,5
X1 і X4	<	>	>	>	>	>	>	>	1,5
X2 і X3	>	<	<	<	<	<	<	<	0,5
X2 і X4	>	>	>	>	>	>	>	>	1,5
X3 і X4	>	>	>	>	>	>	>	>	1,5

Проведемо розрахунок вагомості кожного параметра.

Таблиця 5.5 – Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.		Третя ітер.	
	X1	X2	X3	X4	b_i	$K_{вi}$	b_i^1	$K_{вi}^1$	b_i^2	$K_{вi}^2$
X1	1,0	0,5	0,5	1,5	3,5	0,219	12,25	0,208	42,875	0,212
X2	1,5	1,0	0,5	1,5	4,5	0,281	12,25	0,276	55,125	0,273
X3	1,5	1,5	1,0	1,5	5,5	0,344	21,25	0,36	71,875	0,356
X4	0,5	0,5	0,5	1,0	2,5	0,156	9,25	0,156	32,125	0,159
Всього:					16	1	59	1	202	1

5.5 Аналіз рівня якості варіантів реалізації функцій

У таблиці 5.6 наведемо рівень якості для кожного варіанту реалізації.

Таблиця 5.6 – Показники рівня якості варіантів реалізації

Основні функції	Варіант реалізації функції	Параметри від яких залежить функція	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1(X1)	А	X1	15000	2,5	0,212	0,53
F2(X2)	А	X2	22	1,87	0,273	0,51
F3(X3,X4)	А	X3	490	2,1	0,356	0,735
	А	X4	3000	5	0,159	0,795
	Б	X3	510	1,9	0,356	0,691
	Б	X4	4000	1	0,159	0,159

За даними з таблиці 5.6 визначаємо рівень якості кожного з варіантів:

$$KK1 = 0{,}774 + 0{,}962 + 0{,}835 = 2{,}57,$$

$$KK2 = 0{,}774 + 0{,}962 + 0{,}154 = 1{,}89.$$

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

5.6 Економічний аналіз варіантів розробки ПП

Сформуємо спільні завдання для кожного варіанту реалізації:

1) Створення та розробка проекту ПП;

2) Створення програмної оболонки.

Варіант 1 містить окреме завдання:

3) самостійне аналітичне виведення оцінок параметрів моделей.

Аналогічно варіант 2 містить завдання:

3) використання власних перевірок для обробки даних.

Формула для обчислення загальної трудомісткості:

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М} .$$

Розрахуємо трудомісткість виконання першого завдання. Характеристики: складність алгоритму 1, $T_Р$ =90 людино-днів, $К_П$ = 1.7, $К_{СК}$ = 1. При виконанні першого завдання використовуються вбудовані бібліотеки, тому $К_{СТ}$ = 0.8. Тоді, за формулою (5.4), трудомісткість першого завдання дорівнює:

$$T_1 = 90 \cdot 1{,}7 \cdot 0{,}8 = 122{,}4 \quad людино-днів .$$

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $T_Р$ =27 людино-днів, $К_П$ =0.9,$К_{СК}$ = 1,$К_{СТ}$ =0.8:

$$T_2 = 27 \cdot 0{,}9 \cdot 0{,}8 = 19{,}44 \quad людино-днів .$$

Завдання 3 (використовується алгоритм третьої групи складності, степінь новизни Б):

$$T_3 = 27 \cdot 1{,}12 \cdot 0{,}8 = 24{,}2 \quad людино-днів$$

Завдання 4 (використовується алгоритм третьої групи складності, степінь новизни Б):

$$T_4 = 27 \cdot 1,2 \cdot 0,8 = 26,35 \quad людино-днів$$

Сумарна трудомісткість завдань:

$$T_{\text{I}} = (122,4 + 19,44 + 24,2) \cdot 8 = 1328,64 \quad людино-годин,$$

$$T_{\text{II}} = (122,4 + 19,44 + 26,35) \cdot 8 = 1345,52 \quad людино-годин.$$

Більш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 8000 грн., один фінансовий аналітик з окладом 11000грн. Розрахунок зарплати за годину:

$$C_{ч} = \frac{8000 + 8000 + 11000}{3 \cdot 21 \cdot 8} = 53,57 \quad грн.$$

I. $C_{ЗП} = 53,57 \cdot 1328,64 \cdot 1,2 = 85410,29 \quad грн.,$

II. $C_{ЗП} = 53,57 \cdot 1345,52 \cdot 1,2 = 86495,41 \quad грн..$

Відрахування на єдиний соціальний внесок в залежності від групи професійного ризику (II клас) становить 22%:

I. $C_{ВІД} = C_{ЗП} \cdot 0,22 = 85410,29 \cdot 0,22 = 18790,26 \quad грн.,$

II. $C_{ВІД} = C_{ЗП} \cdot 0,22 = 86495,41 \cdot 0,22 = 19028,99 \quad грн..$

Для однієї машини при коефіцієнту зайнятості 0,2 отримаємо:

$$C_{Г} = 12 \cdot M \cdot K_{З} = 12 \cdot 8000 \cdot 0,2 = 19200 \quad грн.$$

Враховуємо додаткову заробітну плату:

$$C_{ЗП} = C_Г \cdot (1 + K_З) = 19200 \cdot (1 + 0,2) = 23040 \quad грн,$$

$$C_{ВІД} = C_{ЗП} \cdot 0,22 = 23040 \cdot 0,22 = 5068,8 \quad грн.$$

Вважаємо, що амортизація 25%, а вартість ЕОМ – 8000 грн.

$$C_А = K_{ТМ} \cdot K_А \cdot Ц_{ПР} = 1,15 \cdot 0,25 \cdot 8000 = 2300 \quad грн.$$

Витрати на ремонт:

$$C_Р = K_{ТМ} \cdot Ц_{ПР} \cdot K_Р = 1,15 \cdot 8000 \cdot 0,05 = 460 \quad грн.$$

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$T_{ЕФ} = (Д_К - Д_В - Д_С - Д_Р) \cdot t_З \cdot K_В = (365 - 104 - 8 - 16) \cdot 8 \cdot 0,9 = 1706,4 \quad годин$$

Витрати на оплату електроенергії:

$$C_{ЕЛ} = T_{ЕФ} \cdot N_С \cdot K_З \cdot Ц_{ЕН} = 1706,4 \cdot 0,156 \cdot 0,2436 \cdot 1,755 = 113,8 \quad грн.$$

Розрахуємо накладні витрати:

$$C_Н = Ц_{ПР} \cdot 0,67 = 8000 \cdot 0,67 = 5360 \quad грн.$$

Загальна сума експлуатаційний витрат дорівнює:

$$C_{ЕКС} = C_{ЗП} + C_{ВІД} + C_А + C_Р + C_{ЕЛ} + C_Н,$$

$$C_{ЕКС} = 23040 + 5068,8 + 2300 + 460 + 113,8 + 5360 = 36346,6 \quad грн.,$$

$$C_{М-Г} = C_{ЕКС} / T_{ЕФ} = 36346,6 / 1706,4 = 21,3 \quad грн / годину.$$

Розрахуємо витрати на оплату машинного часу для кожного варіанту за формулою:

$$C_M = C_{M-Г} \cdot T,$$

I. $C_M = 21,3 \cdot 1328,64 = 28300,032$ *грн.*,

II. $C_M = 21,3 \cdot 1345,52 = 28659,576$ *грн.*.

Накладні витрати 67% від заробітної плати:

$$C_H = C_{ЗП} \cdot 0,67,$$

I. $C_H = 85410,29 \cdot 0,67 = 57224,89$ *грн.*,

II. $C_H = 86495,41 \cdot 0,67 = 57951,92$ *грн.*.

Загальна вартість розробки ПП для обох варіантів:

$$C_{ПП} = C_{ЗП} + C_{ВІД} + C_M + C_H,$$

I. $C_{ПП} = 85410,29 + 18790,26 + 28300,032 + 57224,89 = 189725,47$ *грн.*,

II. $C_{ПП} = 86495,41 + 19028,99 + 28659,576 + 57951,92 = 192135,89$ *грн.*.

5.7 Вибір кращого варіанта ПП техніко-економічного рівня

Формула для розрахунку коефіцієнта техніко-економічного рівня:

$$K_{ТЕР_j} = K_{К_j} / C_{Ф_j},$$

$$K_{ТЕР_1} = 5,214 / 189725,47 = 0,27 \cdot 10^{-4},$$

$$K_{ТЕР_2} = 3,569 / 192135,89 = 0,19 \cdot 10^{-4}.$$

Для першого варіанту коефіцієнт техніко-економічного рівня $K_{TEP_1} = 0,27 \cdot 10^{-4}$, тому він є більш ефективним, ніж другий варіант.

5.8 Висновки до розділу

Для реалізації програмного продукту кращим варіантом після виконання функціонально-вартісного аналізу виявився перший варіант. Показник техніко-економічного рівня якості для даного варіанту $К_{ТЕР} = 0,27 \cdot 10^{-4}$ і є вищим ніж показник для другого варіанту.

Обраний варіант реалізаціє характеризується такими параметрами:

- мова програмування – C#;
- введення даних файлу;
- для створення інтерфейсу користувача використовується технологія WPF.

Такі характеристики забезпечують швидкодію, зручний інтерфейс та широкий функціонал.

ВИСНОВКИ

В роботі виконано порівняння найбільш вживаних статистичних пакетів: Eviews, SAS та Statistica. Основними напрями роботи при створенні СППР є зручний графічним інтерфейс, введення, редагування та порівняльний аналіз даних; можливість проведення тестів на нестаціонарність.

Розглянуто тести для аналізу нелінійних нестаціонарних процесів. Виконано огляд існуючих методів моделювання і прогнозування стаціонарних та нестаціонарних процесів.

Розглянуто основні критерії якості для оцінювання моделей опису процесів та якості прогнозування. Вибрано критерії: критерій Дарвіна-Уотсона, коефіцієнт детермінації, критерій суми квадратів похибок моделі, інформаційний критерій Акайке; СМПП і СМП для аналізу якості прогнозу. З методів оцінювання параметрів моделі обрано МНК та РМНК.

Розглянуто метод комбінування оцінок як один із варіантів покращення якості моделювання та прогнозування. Встановлено, що якість прогнозів покращується із збільшенням методів прогнозування, але потрібно встановити порогове значення для кількості методів, що використовується.

Застосовано розроблену СППР до аналізу процесів ціноутворення акцій компаній «Укрнафта», «Facebook» та процесу Лоренца.

На основі моделювання і прогнозування визначені кращі моделі для вибраних процесів з використанням методу оцінки параметрів МНК:

- для цін акцій компанії «Укрнафта»– модель АР (1);
- для цін акцій компанії «Facebook» – модель АРКС (10,5);
- для процесу Лоренца – модель АР (8);

Рекомендаціями до подальших досліджень є реалізації інших методів моделювання і прогнозування таких як ARMAX, МГВА і нейронні мережі. Додати реалізацію методу максимальної правдоподібності для оцінювання

параметрів моделі. Додати реалізацію статистичних тестів на стаціонарність. Додати функціональне введення даних, модернізувати інтерфейс.

ЛІТЕРАТУРА

1. Бідюк П. І., Романенко В. Д., Тимощук О. Л. Аналіз часових рядів: навч. посіб. / ННК «Інститут прикладного системного аналізу» Національний технічний університет України «Київський політехнічний інститут», 2010. 317 с.

2. Бідюк П. І. Економетричний аналіз часових рядів. Київ: Політехніка, 2007. 250 с.

3. Молчанов И. Н. Компьютерный практикум по начальному курсу эконометрики (реализация на Eviews): практикум / Ростовский государственный экономический университет. Ростов-н/Д. 2001. 58 с.

4. Ставицький А. В. Навчально-методичний комплекс з курсів «Прогнозування» та «Фінансове прогнозування». Київ: Центр учб. літ., 2006. 107 с.

5. Носко В. П. Эконометрика. Введение в регрессионный анализ временных рядов. Москва: Литкон, 2002. 273 с.

6. Половцев О. В. Системний підхід до моделювання, прогнозування та управління фінансово-економічними процесами. Донецьк: Східний видавничий дім, 2009. 286 с.

7. Лакман И. А., Никульшина Л. М., Шамуратов Н. М. Поддержка принятия решения при выборе пакета обработки статистических данных. Современные проблемы информатизации в экономике и обеспечении безопасности: сборник трудов. Воронеж: Научная книга, 2009. 136 с.

8. Лоусон Ч. К. Численное решение задач методом наименьших квадратов. Москва: Наука, 1986. 234 с.

9. Грешилов А. А. Математические методы построения прогнозов. Москва: Радио и связь, 1997. 112 с.

10. Бідюк П. І. Проектування комп’ютерних інформаційних систем підтримки прийняття рішень: навч. посіб. / ННК «Інститут прикладного системного аналізу» Національний технічний університет України «Київський політехнічний інститут», 2010. 340с.

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

Файл AssessmentMethodInterface.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DSS__Stock_Exchange.AssessmentMethod
{
    interface AssessmentMethodInterface
    {
        double[] teta(double[] Y, double[][] X, double beta, out double[] RS, out double RSS, out double R2, out
double Akaike,out double dw);
    }
}
```

Файл MLS.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DSS__Stock_Exchange.AssessmentMethod
{
    class MLS:AssessmentMethodInterface
    {
        public double[] teta(double[] Y, double[][] X, double beta, out double[] RS, out double RSS, out double
R2, out double Akaike, out double dw)
        {
            Matrix x = new Matrix(X.Length, X[0].Length);
            Matrix y = new Matrix(Y.Length, 1);

            for (int i = 0; i < Y.Length; i++)
            {
                y[i, 0] = Y[i];
                for (int j = 0; j < X[i].Length; j++)
                {
                    x[i, j] = X[i][j];
                }

            }

            Matrix teta = (x.trans() * x).inverse() * (x.trans() * y);
            Matrix ocen_Y = x * teta;

            RS = new double[Y.Length];
            RSS=0;
            dw = 0;
            double TSS=0;
            double med=Y.Average();
            for (int i = 0; i < Y.Length; i++)
            {
                RS[i] = Y[i] - ocen_Y[i, 0];
                RSS += Math.Pow(RS[i], 2);
                TSS += Math.Pow(Y[i] - med, 2);
                if (i > 0) dw += Math.Pow(RS[i] - RS[i - 1], 2);
            }
            R2 = RSS / TSS;
            if (R2 > 1) R2 = 1 / R2;
            R2 = 1 - R2;
            Akaike = Y.Length * Math.Log(RSS) + 2 * (X[0].Length + 1);

            double[] rez = new double[teta.Rows];
            for (int i = 0; i < rez.Length; i++)
            {
                rez[i] = teta[i, 0];
            }
```

```
            dw /= RSS;

            return rez;
        }
    }
}
```

Файл RMLS.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DSS__Stock_Exchange.AssessmentMethod
{
    class RMLS:AssessmentMethodInterface
    {
        public double[] teta(double[] Y, double[][] X, double beta, out double[] RS, out double RSS, out double
R2, out double Akaike, out double dw)
        {
            Matrix teta = new Matrix(X[0].Length, 1);
            Matrix P = new Matrix(X[0].Length, beta);

            for (int i = 0; i < Y.Length; i++)
            {
                Matrix x = new Matrix(X[i].Length, 1);
                for (int j = 0; j < X[i].Length; j++)
                {
                    x[j, 0] = X[i][j];
                }

                teta += (1 / (1 + (x.trans() * P * x)[0, 0])) * P * ((Y[i] - (teta.trans() * x)[0, 0]) * x);
                P += (-1 / (1 + (x.trans() * P * x)[0, 0])) * P * x * x.trans() * P;
            }

            Matrix xxx = new Matrix(X.Length, X[0].Length);
            {
                for (int i = 0; i < Y.Length; i++)
                {
                    for (int j = 0; j < X[i].Length; j++)
                    {
                        xxx[i, j] = X[i][j];
                    }

                }
            }
            Matrix ocen_Y = xxx * teta;

            RS = new double[Y.Length];
            RSS = 0;
            dw = 0;
            double TSS = 0;
            double med = Y.Average();
            for (int i = 0; i < Y.Length; i++)
            {
                RS[i] = Y[i] - ocen_Y[i, 0];
                RSS += Math.Pow(RS[i], 2);
                TSS += Math.Pow(Y[i] - med, 2);
                if (i > 0) dw += Math.Pow(RS[i] - RS[i - 1], 2);
            }
            R2 = RSS / TSS;
            if (R2 > 1) R2 = 1 / R2;
            R2 = 1 - R2;
            Akaike = Y.Length * Math.Log(RSS) + 2 * (X[0].Length + 1);

            double[] rez = new double[teta.Rows];
            for (int i = 0; i < rez.Length; i++)
            {
                rez[i] = teta[i, 0];
            }

            dw /= RSS;
```

```
            return rez;
        }
    }
}
```

Файл AbstractModel.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DSS__Stock_Exchange.Models
{
    public abstract class AbstractModel
    {
        protected double[] a, b;
        protected int[] indexes_a, indexes_b;
        /// <summary>
        /// q==-1 для AP модели
        /// </summary>
        /// <param name="p"></param>
        /// <param name="q"></param>
        protected AbstractModel(int p=-1, int q=-1)
        {
            if (p >= 0)
            {
                indexes_a = new int[p + 1];
                for (int i = 0; i < indexes_a.Length; i++)
                {
                    indexes_a[i] = i;
                }
                a = new double[p + 1];
            }
            if (q >= 0)
            {
                indexes_b = new int[q + 1];
                for (int i = 0; i < indexes_b.Length; i++)
                {
                    indexes_b[i] = i;
                }
                b = new double[q + 1];
            }
            if ((indexes_a == null) && (indexes_b == null)) throw new MyExcepion("Неприпустимі параметри
моделі");
        }

        protected AbstractModel(int[] a = null, int[] b = null)
        {
            if (a != null)
            {
                indexes_a = new int[a.Length];
                this.a = new double[a.Length];
                for (int i = 0; i < indexes_a.Length; i++)
                {
                    indexes_a[i] = a[i];
                }
            }
            if (b != null)
            {
                indexes_b = new int[b.Length];
                this.b = new double[b.Length];
                for (int i = 0; i < indexes_b.Length; i++)
                {
                    indexes_b[i] = b[i];
                }
            }
            if ((indexes_a == null) && (indexes_b == null)) throw new MyExcepion("До моделі не включено жодної
складової");
        }

        public int P
        {
            get
```

```
            {
                if (indexes_a == null) return -1;
                else return indexes_a[indexes_a.Length-1];
            }
        }

        public int Q
        {
            get
            {
                if (indexes_b == null) return -1;
                else return indexes_b[indexes_b.Length-1];
            }
        }

        public abstract int step
        {
            get;
        }

        abstract public double[][] X(double[] Y,double[] MA,int length);
        abstract public double[] Y(double[] Y, int length);
        abstract public void init(double[] teta);
        abstract public double[] forecast_dynamic(double[] Y, double[] MA,int from,int to);
        abstract public double[] forecast_static(double[] Y, double[] MA,int from,int to);
        abstract public string ToString(string y = "y", string ma = "ma");
    }
}
```

Файл ARIMAmodel.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DSS__Stock_Exchange.Models
{
    class ARIMAmodel:AbstractModel
    {
        int d;
        public ARIMAmodel(int p,int d) : base(p)
        {
            if (d>0) this.d = d;
            else throw new MyExcepion("Неприпустимий параметр d");
        }

        public ARIMAmodel(int p,int q,int d)
            : base(p,q)
        {
            if (d > 0) this.d = d;
            else throw new MyExcepion("Неприпустимий параметр d");
            if (q < 0) throw new MyExcepion("Неприпустимий параметр q");
        }

        public ARIMAmodel(int[] a, int[] b,int d)
            : base(a, b)
        {
            if (d > 0) this.d = d;
            else throw new MyExcepion("Неприпустимий параметр d");
        }

        double[] diference(double[] y)
        {
            double[] rez = new double[y.Length];
            for (int i = rez.Length - 1; i > 0; i--)
            {
                rez[i] = y[i] - y[i - 1];
            }
            rez[0] = double.NaN;
            return rez;
        }

        double[] sumar(double y_0, double[] dif)
```

```
        {
            double[] rez = new double[dif.Length];
            rez[0] = y_0 + dif[0];
            for (int i = 1; i < rez.Length; i++)
            {
                rez[i] = rez[i - 1] + dif[i];
            }
            return rez;
        }

        public int D
        {
            get
            {
                return d;
            }
        }

        public override int step
        {
            get { return Math.Max(P, Q) + D; }
        }

        override public double[][] X(double[] y, double[] MA,int length)
        {
            if (MA != null) if (y.Length != MA.Length) throw new Exception();
            if (length <= (Math.Max(P, Q)+D) || (length > y.Length)) throw new MyExcepion("Ряд надто короткий");
            {
                int step = (Math.Max(P, Q) + D);

                var Y = diference(y);
                for (int i = 1; i < d; i++)
                {
                    Y = diference(Y);
                }

                double[][] rez = new double[length-step][];
                for (int i = 0; i < rez.Length; i++)
                {
                    if (indexes_b != null && indexes_a != null) rez[i] = new double[indexes_a.Length +
indexes_b.Length];
                    else if (indexes_a != null) rez[i] = new double[indexes_a.Length];
                    else rez[i] = new double[indexes_b.Length];

                    if (indexes_a != null)
                    {
                        if (indexes_a[0] == 0) rez[i][0] = 1;
                        else rez[i][0] = Y[step + i - indexes_a[0]];

                        for (int j = 1; j < indexes_a.Length; j++)
                        {
                            rez[i][j] = Y[step + i - indexes_a[j]];
                        }
                    }
                    if (indexes_b!=null) for (int j = 0; j < indexes_b.Length; j++)
                    {
                        if (indexes_a != null) rez[i][indexes_a.Length + j] = MA[step + i - indexes_b[j]];
                        else rez[i][j] = MA[step + i - indexes_b[j]];
                    }
                }
                return rez;
            }
        }

        override public double[] Y(double[] y, int length)
        {
            if (length <= (Math.Max(P, Q) + D) || (length > y.Length)) throw new MyExcepion("Ряд надто
короткий");
            int step = (Math.Max(P, Q) + D);

            var Y = diference(y);
            for (int i = 1; i < d; i++)
            {
                Y = diference(Y);
            }
```

```
            double[] rez = new double[length-step];
            for (int i = 0; i < rez.Length; i++)
            {
                rez[i] = Y[step + i];
            }
            return rez;
        }

        override public void init(double[] teta)
        {
            if (indexes_b != null && indexes_a!=null) { if (teta.Length != (a.Length + b.Length)) throw new
Exception(); }
            else if (indexes_a != null) { if (teta.Length != (a.Length)) throw new Exception(); }
            else { if (teta.Length != (b.Length)) throw new Exception(); }

                if (indexes_a != null) for (int i = 0; i < a.Length; i++)
                    {
                        a[i] = teta[i];
                    }

            if (indexes_b!=null) for (int i = 0; i < b.Length; i++)
            {
                if (indexes_a != null) b[i] = teta[a.Length + i];
                else b[i] = teta[i];
            }
        }

        override public double[] forecast_dynamic(double[] y, double[] MA, int from, int to)
        {
            if ((from < step) || (to < step) || (from > y.Length) || (from > to)) throw new
MyExcepion("Неприпустимий діапазон прогнозування");
            else
            {
                if (MA != null) if (y.Length != MA.Length) throw new Exception();

                double[] starts = new double[d];
                starts[0] = y[from - 1];
                double[] Y = diference(y);
                for (int i = 1; i < d; i++)
                {
                    starts[i] = Y[from - 1];
                    Y = diference(Y);
                }

                double[] rez = new double[to - from + 1];
                for (int i = 0; i < rez.Length; i++)
                {
                    if (indexes_a != null)
                    {
                        if (indexes_a[0] == 0) rez[i] = a[0];
                        else if (i - indexes_a[0] >= 0) rez[i] = a[0] * rez[i - indexes_a[0]];
                        else rez[i] = a[0] * Y[from + i - indexes_a[0]];
                        for (int j = 1; j < indexes_a.Length; j++)
                        {
                            if (i - indexes_a[j] >= 0) rez[i] += a[j] * rez[i - indexes_a[j]];
                            else rez[i] += a[j] * Y[from + i - indexes_a[j]];
                        }
                    }

                    if (indexes_b!=null) for (int j = 0; j < indexes_b.Length; j++)
                    {
                        rez[i] += b[j] * MA[from + i - indexes_b[j]];
                    }
                }

                for (int i = starts.Length - 1; i >= 0; i--)
                {
                    rez = sumar(starts[i], rez);
                }

                return rez;
            }
        }

        override public double[] forecast_static(double[] y, double[] MA, int from, int to)
        {
```

```
            if ((from < step) || (to < step) || (from > y.Length) || (from > to)) throw new
MyExcepion("Неприпустимий діапазон прогнозування");
            else
            {
                if (MA!=null) if (y.Length != MA.Length) throw new Exception();

                double[] starts = new double[d];
                starts[0] = y[from - 1];
                double[] Y = diference(y);
                for (int i = 1; i < d; i++)
                {
                    starts[i] = Y[from - 1];
                    Y = diference(Y);
                }

                double[] rez = new double[to - from + 1];
                for (int i = 0; i < rez.Length; i++)
                {
                    if (indexes_a != null)
                    {
                        if (indexes_a[0] == 0) rez[i] = a[0];
                        else rez[i] = a[0] * Y[from + i - indexes_a[0]];
                        for (int j = 1; j < indexes_a.Length; j++)
                        {
                            rez[i] += a[j] * Y[from + i - indexes_a[j]];
                        }
                    }
                    if (indexes_b != null) for (int j = 0; j < indexes_b.Length; j++)
                        {
                            rez[i] += b[j] * MA[from + i - indexes_b[j]];
                        }
                }

                for (int i = starts.Length - 1; i >= 0; i--)
                {
                    rez = sumar(starts[i], rez);
                }

                return rez;
            }
        }

        override public string ToString(string y="y",string ma="eps")
        {
            string rez = "d^" + d + " " + y + "(k)=";

            if (indexes_a != null)
            {
                if (indexes_a[0] == 0) rez += a[0];
                else rez += a[0] + "*d^" + d + " " +y+"(k-" + indexes_a[0] + ")";
                for (int j = 1; j < indexes_a.Length; j++)
                {
                    rez += "+" + a[j] + "*d^" + d + " " +y+"(k-" + indexes_a[j] + ")";
                }
                if (indexes_b != null) rez += "+";
            }

            if (indexes_b!=null) for (int j = 0; j < indexes_b.Length; j++)
            {
                if (j > 0) rez += "+";
                if (indexes_b[j]!=0) rez += b[j] + "*"+ma+"(k-" + indexes_b[j] + ")";
                else rez += b[j] + "*"+ma+"(k)";
            }

            return rez;
        }
    }
}
```

Файл ARMAmodel.cs

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DSS__Stock_Exchange.Models
{
    public class ARMAmodel:AbstractModel
    {
        public ARMAmodel(int p) : base(p)
        {

        }

        public ARMAmodel(int p,int q)
            : base(p,q)
        {
            if (q < 0) throw new MyExcepion("Неприпустимий параметр q");
        }

        public ARMAmodel(int[] a, int[] b)
            : base(a, b)
        {

        }

        public override int step
        {
            get { return Math.Max(P, Q); }
        }

        override public double[][] X(double[] Y, double[] MA,int length)
        {
            if (MA != null) if (Y.Length != MA.Length) throw new Exception();
            if (length <= Math.Max(P, Q) || (length > Y.Length)) throw new MyExcepion("Ряд надто короткий");
            {
                int step = Math.Max(P, Q);
                double[][] rez = new double[length-step][];
                for (int i = 0; i < rez.Length; i++)
                {
                    if (indexes_b != null && indexes_a != null) rez[i] = new double[indexes_a.Length +
indexes_b.Length];
                    else if (indexes_a != null) rez[i] = new double[indexes_a.Length];
                    else rez[i] = new double[indexes_b.Length];

                    if (indexes_a != null)
                    {
                        if (indexes_a[0] == 0) rez[i][0] = 1;
                        else rez[i][0] = Y[step + i - indexes_a[0]];

                        for (int j = 1; j < indexes_a.Length; j++)
                        {
                            rez[i][j] = Y[step + i - indexes_a[j]];
                        }
                    }
                    if (indexes_b!=null) for (int j = 0; j < indexes_b.Length; j++)
                    {
                        if (indexes_a != null) rez[i][indexes_a.Length + j] = MA[step + i - indexes_b[j]];
                        else rez[i][j] = MA[step + i - indexes_b[j]];
                    }
                }
                return rez;
            }
        }

        override public double[] Y(double[] Y, int length)
        {
            if (length <= Math.Max(P, Q) || (length > Y.Length)) throw new MyExcepion("Ряд надто короткий");
            int step = Math.Max(P, Q);
            double[] rez = new double[length-step];
            for (int i = 0; i < rez.Length; i++)
            {
                rez[i] = Y[step + i];
            }
            return rez;
        }
```

```
        override public void init(double[] teta)
        {
            if (indexes_b != null && indexes_a!=null) { if (teta.Length != (a.Length + b.Length)) throw new
Exception(); }
            else if (indexes_a != null) { if (teta.Length != (a.Length)) throw new Exception(); }
            else { if (teta.Length != (b.Length)) throw new Exception(); }

                if (indexes_a != null) for (int i = 0; i < a.Length; i++)
                    {
                        a[i] = teta[i];
                    }

                if (indexes_b!=null) for (int i = 0; i < b.Length; i++)
                {
                    if (indexes_a != null) b[i] = teta[a.Length + i];
                    else b[i] = teta[i];
                }
        }

        override public double[] forecast_dynamic(double[] Y, double[] MA, int from, int to)
        {
            if ((from < Math.Max(P, Q)) || (to < Math.Max(P, Q)) || (from > Y.Length) || (from > to)) throw new
MyExcepion("Неприпустимий діапазон прогнозування");
            else
            {
                if (MA != null) if (Y.Length != MA.Length) throw new Exception();
                double[] rez = new double[to - from + 1];
                for (int i = 0; i < rez.Length; i++)
                {
                    if (indexes_a != null)
                    {
                        if (indexes_a[0] == 0) rez[i] = a[0];
                        else if (i - indexes_a[0] >= 0) rez[i] = a[0] * rez[i - indexes_a[0]];
                        else rez[i] = a[0] * Y[from + i - indexes_a[0]];
                        for (int j = 1; j < indexes_a.Length; j++)
                        {
                            if (i - indexes_a[j] >= 0) rez[i] += a[j] * rez[i - indexes_a[j]];
                            else rez[i] += a[j] * Y[from + i - indexes_a[j]];
                        }
                    }

                    if (indexes_b!=null) for (int j = 0; j < indexes_b.Length; j++)
                    {
                        rez[i] += b[j] * MA[from + i - indexes_b[j]];
                    }
                }
                return rez;
            }
        }

        override public double[] forecast_static(double[] Y, double[] MA, int from, int to)
        {
            if ((from < Math.Max(P, Q)) || (to < Math.Max(P, Q)) || (from > Y.Length) || (from > to)) throw new
MyExcepion("Неприпустимий діапазон прогнозування");
            else
            {
                if (MA!=null) if (Y.Length != MA.Length) throw new Exception();
                double[] rez = new double[to - from + 1];
                for (int i = 0; i < rez.Length; i++)
                {
                    if (indexes_a != null)
                    {
                        if (indexes_a[0] == 0) rez[i] = a[0];
                        else rez[i] = a[0] * Y[from + i - indexes_a[0]];
                        for (int j = 1; j < indexes_a.Length; j++)
                        {
                            rez[i] += a[j] * Y[from + i - indexes_a[j]];
                        }
                    }
                    if (indexes_b != null) for (int j = 0; j < indexes_b.Length; j++)
                        {
                            rez[i] += b[j] * MA[from + i - indexes_b[j]];
                        }
                }
                return rez;
            }
```

```
        }

        override public string ToString(string y="y",string ma="eps")
        {
            string rez = y+"(k)=";

            if (indexes_a != null)
            {
                if (indexes_a[0] == 0) rez += a[0];
                else rez += a[0] + "*"+y+"(k-" + indexes_a[0] + ")";
                for (int j = 1; j < indexes_a.Length; j++)
                {
                    rez += "+" + a[j] + "*"+y+"(k-" + indexes_a[j] + ")";
                }
                if (indexes_b != null) rez += "+";
            }

            if (indexes_b!=null) for (int j = 0; j < indexes_b.Length; j++)
            {
                if (j > 0) rez += "+";
                if (indexes_b[j]!=0) rez += b[j] + "*"+ma+"(k-" + indexes_b[j] + ")";
                else rez += b[j] + "*"+ma+"(k)";
            }

            return rez;
        }
    }
}
```

Файл MovingAverageInterface.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DSS__Stock_Exchange.MovingAverage
{
    public interface MovingAverageInterface
    {
        double[] ma(double[] x,int size);
    }
}
```

Файл SimpleMovingAverage.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DSS__Stock_Exchange.MovingAverage
{
    class SimpleMovingAverage:MovingAverageInterface
    {
        public double[] ma(double[] x, int size)
        {
            double[] rez = new double[x.Length];
            for (int i = 0; i < rez.Length; i++)
            {
                rez[i] = 0;
                for (int j = 0; j < size; j++)
                {
                    if (i - j >= 0) rez[i] += x[i - j];
                    else break;
                }
                 rez[i] /= size;
            }
            return rez;
        }
    }
}
```

Файл ExponentialMovingAverageDecrease.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DSS__Stock_Exchange.MovingAverage
{
    class ExponentialMovingAverageDecrease:MovingAverageInterface
    {
        public double[] ma(double[] x, int size)
        {
            double alp = (2.0 / (size + 1));
            double[] w = new double[size];
            for (int i = 0; i < w.Length; i++)
            {
                w[i] = Math.Pow(1-alp,i+1);
            }

            double sum = w.Sum();
            double[] rez = new double[x.Length];
            for (int i = 0; i < rez.Length; i++)
            {
                rez[i] = 0;
                for (int j = 0; j < w.Length; j++)
                {
                    if (i - j >= 0) rez[i] += w[j] * x[i - j];
                    else break;
                }
                rez[i] /= sum;
            }
            return rez;
        }
    }
}
```

Файл ExponentialMovingAverageIncrease.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DSS__Stock_Exchange.MovingAverage
{
    class ExponentialMovingAverageIncrease:MovingAverageInterface
    {
        public double[] ma(double[] x, int size)
        {
            double alp = (2.0 / (size + 1));
            double[] w = new double[size];
            for (int i = 0; i < w.Length; i++)
            {
                w[i] = Math.Pow(1 - alp, size - i);
            }

            double sum = w.Sum();
            double[] rez = new double[x.Length];
            for (int i = 0; i < rez.Length; i++)
            {
                rez[i] = 0;
                for (int j = 0; j < w.Length; j++)
                {
                    if (i - j >= 0) rez[i] += w[j] * x[i - j];
                    else break;
                }
                rez[i] /= sum;
            }
            return rez;
        }
    }
}
```

Файл MainWindow.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace DSS__Stock_Exchange
{
    /// <summary>
    /// Логика взаимодействия для MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            TimeSeries.DataContext = TimeSeriesClass.dataTable.DefaultView;
        }

        private void MenuItem_Click_Exit(object sender, RoutedEventArgs e)
        {
            Environment.Exit(0);
        }

        private void MenuItem_Click_Load_TimeSeries(object sender, RoutedEventArgs e)
        {
            try
            {
                Microsoft.Win32.OpenFileDialog ofd = new Microsoft.Win32.OpenFileDialog();
                ofd.Filter = "Text Files (*.txt)|*.txt|All Files (*.*)|*.*";
                if (ofd.ShowDialog() == true)
                {
                    System.IO.StreamReader reader = new System.IO.StreamReader(ofd.FileName);
                    List<double> series = new List<double>();
                    string s;
                    while ((s = reader.ReadLine()) != null)
                    {
                        series.Add(double.Parse(s));
                    }
                    reader.Close();
                    (new InputSeriesNameWindow(series.ToArray())).Show();
                }
            }
            catch
            {
                MessageBox.Show("Не вдалося відкрити файл","Помилка");
            }
        }

        private void MenuItem_Click_Add_Empty_TimeSeries(object sender, RoutedEventArgs e)
        {
            (new InputSeriesSizeWindow()).Show();
        }

        private void MenuItem_Click_Copy_TimeSeries(object sender, RoutedEventArgs e)
        {
            var SelectedItems = TimeSeries.SelectedItems;
            for (int i = 0; i < SelectedItems.Count; i++)
            {
                (new
InputSeriesNameWindow(TimeSeriesClass.ReturnRow(TimeSeries.Items.IndexOf(SelectedItems[i])))).Show();
            }
        }
```

```
private void MenuItem_Click_Delete_TimeSeries(object sender, RoutedEventArgs e)
{
    int SelectedIndex = TimeSeries.SelectedIndex;
    while (SelectedIndex >= 0)
    {
        TimeSeriesClass.DelDataAr(SelectedIndex);
        SelectedIndex = TimeSeries.SelectedIndex;
    }
}

private void MenuItem_Click_Show_TimeSeries(object sender, RoutedEventArgs e)
{
    var SelectedItems = TimeSeries.SelectedItems;
    int[] SelectedIndexes = new int[SelectedItems.Count];
    for (int i = 0; i < SelectedIndexes.Length; i++)
    {
        SelectedIndexes[i]=TimeSeries.Items.IndexOf(SelectedItems[i]);
    }
    if (SelectedIndexes.Length>0) (new TimeSeriesShowWindow(SelectedIndexes)).Show();
}

private void MenuItem_Click_Save_TimeSeries(object sender, RoutedEventArgs e)
{
    Microsoft.Win32.OpenFileDialog ofd = new Microsoft.Win32.OpenFileDialog();
    ofd.Filter = "Text Files (*.txt)|*.txt|All Files (*.*)|*.*";
    if (ofd.ShowDialog() == true)
    {
        System.IO.StreamWriter writer = new System.IO.StreamWriter(ofd.FileName);

        var SelectedItems = TimeSeries.SelectedItems;
        for (int i = 0; i < SelectedItems.Count; i++)
        {
            double[] serie = TimeSeriesClass.ReturnRow(TimeSeries.Items.IndexOf(SelectedItems[i]));
            writer.WriteLine(TimeSeriesClass.NameRow(TimeSeries.Items.IndexOf(SelectedItems[i])));
            for (int j = 0; j < serie.Length; j++)
            {
                writer.WriteLine(serie[j]);
            }
            writer.WriteLine();
        }

        writer.Close();
    }
}

private void MenuItem_Click_Show_Statistic(object sender, RoutedEventArgs e)
{
    var SelectedItems = TimeSeries.SelectedItems;
    double[][] ryadi= new double[SelectedItems.Count][];
    string[] names=new string[SelectedItems.Count];
    for (int i = 0; i < ryadi.Length; i++)
    {
        ryadi[i]=TimeSeriesClass.ReturnRow(TimeSeries.Items.IndexOf(SelectedItems[i]));
        names[i]=TimeSeriesClass.NameRow(TimeSeries.Items.IndexOf(SelectedItems[i]));
    }
    Statistics st = new Statistics(ryadi);
    double[] aver = st.MathAverages();
    double[] disp = st.Variances();
    double[] asym = st.Skewnesses();
    double[] eks = st.Kurtosises();

    for (int i = 0; i < names.Length; i++)
    {
        (new ShowStatisticWindow(aver[i], disp[i], asym[i], eks[i], names[i])).Show();
    }
}

private void MenuItem_Click_Show_Correlation(object sender, RoutedEventArgs e)
{
    try
    {
        var SelectedItems = TimeSeries.SelectedItems;
        double[][] ryadi = new double[SelectedItems.Count][];
        string[] names = new string[SelectedItems.Count];
        for (int i = 0; i < ryadi.Length; i++)
```

```
                {
                    ryadi[i] = TimeSeriesClass.ReturnRow(TimeSeries.Items.IndexOf(SelectedItems[i]));
                    names[i] = TimeSeriesClass.NameRow(TimeSeries.Items.IndexOf(SelectedItems[i]));
                }
                Statistics st = new Statistics(ryadi);
                (new ShowCorelationWindow(st.Kovariations(), st.Korelations(),names)).Show();
            }
            catch
            {
                MessageBox.Show("Ряди мають різну розмірність", "Помилка");
            }
        }

        private void MenuItem_Click_Build_Model(object sender, RoutedEventArgs e)
        {
            var menuitem = sender as MenuItem;
            if (string.Equals(menuitem.Header,"АРКС")) (new ModelWindow(ModelType.ARMA)).Show();
            else if (string.Equals(menuitem.Header, "АРІКС")) (new ModelWindow(ModelType.ARIMA)).Show();
        }

        private void MenuItem_Click_Show_CorrelationFunctions(object sender, RoutedEventArgs e)
        {
            var SelectedItems = TimeSeries.SelectedItems;
            double[][] ryadi = new double[SelectedItems.Count][];
            string[] names = new string[SelectedItems.Count];
            for (int i = 0; i < ryadi.Length; i++)
            {
                ryadi[i] = TimeSeriesClass.ReturnRow(TimeSeries.Items.IndexOf(SelectedItems[i]));
                names[i] = TimeSeriesClass.NameRow(TimeSeries.Items.IndexOf(SelectedItems[i]));
            }
            Statistics st = new Statistics(ryadi);

            if (SelectedItems.Count>0) (new InputLagWindow(st, names)).Show();
        }

        private void MenuItem_Click_Build_MovingAverage(object sender, RoutedEventArgs e)
        {
            var menuitem = sender as MenuItem;
            MovingAverage.MovingAverageInterface ma;
            if (string.Equals(menuitem.Header, "Побудувати просте ковзнє середнє"))
            {
                ma = new MovingAverage.SimpleMovingAverage();
            }
            else if (string.Equals(menuitem.Header, "Побудувати експоненційне ковзнє середнє (з акцентом на
останні виміри)"))
            {
                ma = new MovingAverage.ExponentialMovingAverageDecrease();
            }else
            {
                ma = new MovingAverage.ExponentialMovingAverageIncrease();
            }

            var SelectedItems = TimeSeries.SelectedItems;
            for (int i = 0; i < SelectedItems.Count; i++)
            {
                (new
AddMovingAverageWindow(menuitem.Header.ToString(),TimeSeries.Items.IndexOf(SelectedItems[i]),ma)).Show();
            }
        }

        private void Window_Closed(object sender, EventArgs e)
        {
            Environment.Exit(0);
        }
    }
}
```

Файл ModelType.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace DSS__Stock_Exchange
{
    public enum ModelType { ARMA, ARIMA}
}
```

Файл ModelWindow.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using DSS__Stock_Exchange.AssessmentMethod;
using DSS__Stock_Exchange.Models;

namespace DSS__Stock_Exchange
{
    /// <summary>
    /// Логика взаимодействия для ARCHmodelWindow.xaml
    /// </summary>
    public partial class ModelWindow : Window
    {
        ModelType mt;
        public ModelWindow(ModelType mt)
        {
            try
            {
                InitializeComponent();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
            this.mt = mt;
            switch (mt)
            {
                case ModelType.ARMA:
                    Title += "APKC";
                    break;
                case ModelType.ARIMA:
                    Title += "APIKC";
                    GroupBox_d.Visibility = System.Windows.Visibility.Visible;
                    break;
            }

            ComboBox_Chose_Series.ItemsSource = TimeSeriesClass.RowNames;

            {
                string[] types_meth = { "MHK", "PMHK"};
                ComboBox_Method.ItemsSource = types_meth;
                ComboBox_Method.SelectedIndex = 0;
            }
        }

        private void TextBox_Count_KeyDown(object sender, KeyEventArgs e)
        {
            Key[] keys = { Key.D0, Key.D1, Key.D2, Key.D3, Key.D4, Key.D5, Key.D6, Key.D7, Key.D8, Key.D9 };
            if (!keys.Contains(e.Key)) e.Handled = true;
        }

        private void ComboBox_Chose_Series_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            try
            {
                TextBox_Count.Text =
TimeSeriesClass.ReturnRow(ComboBox_Chose_Series.SelectedIndex).Length.ToString();
```

```
            Button_Build.IsEnabled = true;
        }
        catch
        {

        }
    }

    private void CheckBox_MA_Checked(object sender, RoutedEventArgs e)
    {
        if (GroupBox_MA!=null) GroupBox_MA.IsEnabled = true;

        if (TextBox_Q_regresors!=null) TextBox_Q_regresors.Text = "0 1 2 3";
    }

    private void CheckBox_MA_Unchecked(object sender, RoutedEventArgs e)
    {
        if (GroupBox_MA != null) GroupBox_MA.IsEnabled = false;
        CheckBox_AR.IsChecked = true;

        if (TextBox_Q_regresors != null) TextBox_Q_regresors.Text = "";
    }

    private void Button_Click(object sender, RoutedEventArgs e)
    {
        try
        {
            double[] Y = TimeSeriesClass.ReturnRow(ComboBox_Chose_Series.SelectedItem.ToString());
            AssessmentMethodInterface method;
            switch (ComboBox_Method.SelectedIndex)
            {
                case 0:
                    method = new MLS();
                    break;
                default:
                    method = new RMLS();
                    break;

            }
            double[] ma=null;
            AbstractModel model;
            if ((bool)CheckBox_details.IsChecked)
            {
                int[] a=null,b=null;

                if (!string.Equals(TextBox_P_regresors.Text,""))
                {
                    string[] s_a = TextBox_P_regresors.Text.Split(' ');
                    a = new int[s_a.Length];
                    for (int i = 0; i < a.Length; i++)
                    {
                        a[i] = int.Parse(s_a[i]);
                    }
                }
                else
                {
                    a = null;
                    MessageBox.Show("Не задано жодної складової авторегресії", "Попередження");
                }

                if (!string.Equals(TextBox_Q_regresors.Text,""))
                {
                    string[] s_b = TextBox_Q_regresors.Text.Split(' ');
                    b = new int[s_b.Length];
                    for (int i = 0; i < b.Length; i++)
                    {
                        b[i] = int.Parse(s_b[i]);
                    }
                }
                else
                {
                    b = null;
                    MessageBox.Show("Не задано жодної складової ковзного середнього","Попередження");
                }

                if (b != null)
```

```
                    {
                        if ((bool)RadioButton_build_ma.IsChecked)
                        {
                            ma = new double[Y.Length];
                            White_Noise.Norm(ma, 0, 1);
                        }
                        else
                        {
                            ma = TimeSeriesClass.ReturnRow(ComboBox_TimeSeries_MA.SelectedItem.ToString());
                        }
                    }

                    switch (mt)
                    {
                        case ModelType.ARMA:
                            model = new ARMAmodel(a, b);
                            break;
                        default:
                            model = new ARIMAmodel(a, b, int.Parse(TextBox_d_param.Text));
                            break;
                    }

                }
                else
                {
                    if ((bool)CheckBox_MA.IsChecked)
                    {
                        if ((bool)CheckBox_AR.IsChecked)
                        {
                            switch (mt)
                            {
                                case ModelType.ARMA:
                                    model = new ARMAmodel(int.Parse(TextBox_P_param.Text),
int.Parse(TextBox_Q_param.Text));
                                    break;
                                default:
                                    model = new ARIMAmodel(int.Parse(TextBox_P_param.Text),
int.Parse(TextBox_Q_param.Text), int.Parse(TextBox_d_param.Text));
                                    break;
                            }
                        }
                        else
                        {
                            switch (mt)
                            {
                                case ModelType.ARMA:
                                    model = new ARMAmodel(0, int.Parse(TextBox_Q_param.Text));
                                    break;
                                default:
                                    model = new ARIMAmodel(0, int.Parse(TextBox_Q_param.Text),
int.Parse(TextBox_d_param.Text));
                                    break;
                            }
                        }
                        if ((bool)RadioButton_build_ma.IsChecked)
                        {
                            ma = new double[Y.Length];
                            White_Noise.Norm(ma, 0, 1);
                        }
                        else
                        {
                            ma = TimeSeriesClass.ReturnRow(ComboBox_TimeSeries_MA.SelectedItem.ToString());
                        }
                    }
                    else
                    {
                        switch (mt)
                        {
                            case ModelType.ARMA:
                                model = new ARMAmodel(int.Parse(TextBox_P_param.Text));
                                break;
                            default:
                                model = new ARIMAmodel(int.Parse(TextBox_P_param.Text),
int.Parse(TextBox_d_param.Text));
                                break;
                        }
```

```
                    }
                }

                double[] RS;
                double RSS, R2, Akaike,dw;
                model.init(method.teta(model.Y(Y, int.Parse(TextBox_Count.Text)), model.X(Y, ma,
int.Parse(TextBox_Count.Text)), double.Parse(TextBox_Speed.Text), out RS, out RSS, out R2, out Akaike,out dw));

                if ((bool)CheckBox_SaveResid.IsChecked)
                {
                    int step = Math.Max(model.P, model.Q);
                    double[] tmp = new double[RS.Length + step];
                    for (int i = 0; i < tmp.Length; i++)
                    {
                        if (i < step) tmp[i] = double.NaN;
                        else tmp[i] = RS[i - step];
                    }
                    TimeSeriesClass.AddDataAr(tmp, "Похибки " +
TimeSeriesClass.NameRow(ComboBox_Chose_Series.SelectedIndex));
                }

                (new
ShowModelWindow(TimeSeriesClass.NameRow(ComboBox_Chose_Series.SelectedIndex),RSS,R2,Akaike,dw,model,Y,ma,mt)).Sh
ow();
            }
            catch (MyExcepion ex)
            {
                MessageBox.Show(ex.Message,"Помилка");
            }
            catch
            {
                MessageBox.Show("Некоректно задані параметри","Помилка");
            }
        }

        private void CheckBox_AR_Checked(object sender, RoutedEventArgs e)
        {
            if (GroupBox_AR != null) GroupBox_AR.IsEnabled = true;

            if (TextBox_P_regresors != null) TextBox_P_regresors.Text = "0 1 2 3";
        }

        private void CheckBox_AR_Unchecked(object sender, RoutedEventArgs e)
        {
            if (GroupBox_AR != null) GroupBox_AR.IsEnabled = false;
            CheckBox_MA.IsChecked = true;

            if (TextBox_P_regresors != null)  TextBox_P_regresors.Text = "";
        }

        private void ComboBox_Method_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            if (ComboBox_Method.SelectedIndex == 1)
            {
                if (Label_Speed != null) Label_Speed.Visibility = System.Windows.Visibility.Visible;
                if (TextBox_Speed != null) TextBox_Speed.Visibility = System.Windows.Visibility.Visible;
            }
            else
            {
                if (Label_Speed != null) Label_Speed.Visibility = System.Windows.Visibility.Hidden;
                if (TextBox_Speed != null) TextBox_Speed.Visibility = System.Windows.Visibility.Hidden;
            }
        }

        private void RadioButton_build_ma_Checked(object sender, RoutedEventArgs e)
        {
            if (RadioButton_load_ma!=null) RadioButton_load_ma.IsChecked = false;
        }

        private void RadioButton_build_ma_Unchecked(object sender, RoutedEventArgs e)
        {
        }

        private void RadioButton_load_ma_Checked(object sender, RoutedEventArgs e)
        {
```

```
            if (RadioButton_build_ma != null) RadioButton_build_ma.IsChecked = false;
            if (Grid_MA_load != null)
            {
                ComboBox_TimeSeries_MA.ItemsSource = TimeSeriesClass.RowNames;
                Grid_MA_load.Visibility = System.Windows.Visibility.Visible;
            }
        }

        private void RadioButton_load_ma_Unchecked(object sender, RoutedEventArgs e)
        {
            if (Grid_MA_load != null) Grid_MA_load.Visibility = System.Windows.Visibility.Hidden;
        }

        private void TextBox_P_regresors_KeyDown(object sender, KeyEventArgs e)
        {
            Key[] keys = { Key.D0, Key.D1, Key.D2, Key.D3, Key.D4, Key.D5, Key.D6, Key.D7, Key.D8, Key.D9,
Key.Space };
            if (!keys.Contains(e.Key)) e.Handled = true;
        }

        private void CheckBox_details_Checked(object sender, RoutedEventArgs e)
        {
            if (Grid_simple_p != null) Grid_simple_p.Visibility = System.Windows.Visibility.Hidden;
            if (Grid_detail_p != null) Grid_detail_p.Visibility = System.Windows.Visibility.Visible;

            if (Grid_simple_q != null) Grid_simple_q.Visibility = System.Windows.Visibility.Hidden;
            if (Grid_detail_q != null) Grid_detail_q.Visibility = System.Windows.Visibility.Visible;
        }

        private void CheckBox_details_Unchecked(object sender, RoutedEventArgs e)
        {
            if (Grid_simple_p != null) Grid_simple_p.Visibility = System.Windows.Visibility.Visible;
            if (Grid_detail_p != null) Grid_detail_p.Visibility = System.Windows.Visibility.Hidden;

            if (Grid_simple_q != null) Grid_simple_q.Visibility = System.Windows.Visibility.Visible;
            if (Grid_detail_q != null) Grid_detail_q.Visibility = System.Windows.Visibility.Hidden;
        }

        private void TextBox_Q_regresors_KeyDown(object sender, KeyEventArgs e)
        {
            Key[] keys = { Key.D0, Key.D1, Key.D2, Key.D3, Key.D4, Key.D5, Key.D6, Key.D7, Key.D8, Key.D9,
Key.Space };
            if (!keys.Contains(e.Key)) e.Handled = true;
        }

        private void TextBox_P_regresors_LostFocus(object sender, RoutedEventArgs e)
        {
            string s=TextBox_P_regresors.Text;
            if (s.Length > 0)
            {
                while (s.Length > 0 && s[0] == ' ')
                {
                    s = s.Remove(0, 1);
                }
                for (int i = 1; i < s.Length; i++)
                {
                    if (s[i] == ' ')
                    {
                        if (i + 1 < s.Length) while (s[i + 1] == ' ')
                        {
                            s = s.Remove(i, 1);
                            if (i + 1 >= s.Length) break;
                        }
                    }
                }
                if (s.Length>0) if (s[s.Length - 1] == ' ') s = s.Remove(s.Length - 1,1);
                TextBox_P_regresors.Text = s;
            }
        }

        private void TextBox_Q_regresors_LostFocus(object sender, RoutedEventArgs e)
        {
            string s = TextBox_Q_regresors.Text;
            if (s.Length > 0)
            {
                while (s.Length > 0 && s[0] == ' ')
```

```
                {
                    s = s.Remove(0, 1);
                }
                for (int i = 1; i < s.Length; i++)
                {
                    if (s[i] == ' ')
                    {
                        if (i + 1 < s.Length) while (s[i + 1] == ' ')
                            {
                                s = s.Remove(i, 1);
                                if (i + 1 >= s.Length) break;
                            }
                    }
                }
                if (s.Length > 0) if (s[s.Length - 1] == ' ') s = s.Remove(s.Length - 1, 1);
                TextBox_Q_regresors.Text = s;
            }
        }
    }
}
```

Файл Statistics.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using DSS__Stock_Exchange.AssessmentMethod;

namespace DSS__Stock_Exchange
{
    public class Statistics
    {
        double[][] TimeSeriesArray;
        public Statistics(double[][] TimeSeriesArray)
        {
            this.TimeSeriesArray = new double[TimeSeriesArray.Length][];
            for (int i = 0; i < this.TimeSeriesArray.Length; i++)
            {
                this.TimeSeriesArray[i] = new double[TimeSeriesArray[i].Length];
                for (int j = 0; j < this.TimeSeriesArray[i].Length; j++)
                {
                    this.TimeSeriesArray[i][j] = TimeSeriesArray[i][j];
                }
            }
        }

        public double[] MathAverages()
        {
            double[] rez = new double[TimeSeriesArray.Length];
            for (int i = 0; i < rez.Length; i++) rez[i] = TimeSeriesArray[i].Average();
            return rez;
        }

        public double[] Variances()
        {
            double[] rez = new double[TimeSeriesArray.Length];
            for (int i = 0; i < rez.Length; i++)
            {
                double aver = TimeSeriesArray[i].Average();
                rez[i] = 0;
                for (int j = 0; j < TimeSeriesArray[i].Length; j++)
                {
                    rez[i] += Math.Pow(TimeSeriesArray[i][j]-aver, 2);
                }
                rez[i] /= TimeSeriesArray[i].Length - 1;
            }
            return rez;
        }

        public double[] Skewnesses()
        {
            double[] rez = new double[TimeSeriesArray.Length];
            double[] disp = Variances();
```

```
    for (int i = 0; i < rez.Length; i++)
    {
        double aver = TimeSeriesArray[i].Average();
        rez[i] = 0;
        for (int j = 0; j < TimeSeriesArray[i].Length; j++)
        {
            rez[i] += Math.Pow(TimeSeriesArray[i][j] - aver, 3);
        }
        rez[i] /= TimeSeriesArray[i].Length;
        rez[i] /= Math.Pow(Math.Sqrt(disp[i]), 3);
    }
    return rez;
}

public double[] Kurtosises()
{
    double[] rez = new double[TimeSeriesArray.Length];
    double[] disp = Variances();
    for (int i = 0; i < rez.Length; i++)
    {
        double aver = TimeSeriesArray[i].Average();
        rez[i] = 0;
        for (int j = 0; j < TimeSeriesArray[i].Length; j++)
        {
            rez[i] += Math.Pow(TimeSeriesArray[i][j] - aver, 4);
        }
        rez[i] /= TimeSeriesArray[i].Length;
        rez[i] /= Math.Pow(Math.Sqrt(disp[i]), 4);
        //rez[i] -= 3;
    }
    return rez;
}

public double[][] Kovariations()
{
    for (int i = 0; i < TimeSeriesArray.Length - 1; i++)
    {
        if (TimeSeriesArray[i].Length != TimeSeriesArray[i + 1].Length) throw new Exception();
    }

    double[][] rez = new double[TimeSeriesArray.Length][];
    for (int i = 0; i < rez.Length; i++)
    {
        rez[i] = new double[TimeSeriesArray.Length];
        for (int j = 0; j < rez[i].Length; j++)
        {
            double aver_i = TimeSeriesArray[i].Average(), aver_j = TimeSeriesArray[j].Average();
            rez[i][j] = 0;
            for (int k = 0; k < TimeSeriesArray[i].Length; k++)
            {
                rez[i][j] += (TimeSeriesArray[i][k] - aver_i) * (TimeSeriesArray[j][k] - aver_j);
            }
            rez[i][j] /= TimeSeriesArray[i].Length - 1;
        }
    }

    return rez;
}

public double[][] Korelations()
{
    double[][] rez = Kovariations();
    double[] disp = Variances();
    for (int i = 0; i < rez.Length; i++)
    {
        for (int j = 0; j < rez[i].Length; j++)
        {
            rez[i][j] /= Math.Sqrt(disp[i]) * Math.Sqrt(disp[j]);
        }
    }
    return rez;
}

public double[][] ACF(int lag)
{
    double[][] rez=new double[TimeSeriesArray.Length][];
```

```
            double[] ser = MathAverages();
            double[] disp = Variances();

            for (int i = 0; i < rez.Length; i++)
            {
                rez[i] = new double[lag];
                for (int j = 0; j < rez[i].Length; j++)
                {
                    rez[i][j] = 0;
                    for (int k = j + 1; k < TimeSeriesArray[i].Length; k++)
                    {
                        rez[i][j] += (TimeSeriesArray[i][k] - ser[i]) * (TimeSeriesArray[i][k - (j + 1)] -
ser[i]);
                    }
                    rez[i][j] /= disp[i];
                    rez[i][j] /= TimeSeriesArray[i].Length - 1;
                }
            }

            return rez;
        }

        public double[][] PACF(int lag)
        {
            double[][] rez = new double[TimeSeriesArray.Length][];
            double[][] acf = ACF(lag);

            for (int i = 0; i < rez.Length; i++)
            {
                rez[i] = new double[lag];
                for (int j = 0; j < rez[i].Length; j++)
                {
                    Matrix matr = new Matrix(j+1);
                    for (int k = 0; k < j; k++)
                    {
                        for (int p = 0; p < j-k; p++)
                        {
                            matr[k, k + p + 1] = acf[i][p];
                            matr[k + p + 1, k] = acf[i][p];
                        }
                    }

                    Matrix vect = new Matrix(j+1,1);
                    for (int k = 0; k < j+1; k++)
                    {
                        vect[k, 0] = acf[i][k];
                    }
                    rez[i][j] = (matr.inverse()*vect)[j,0];
                }
            }

            return rez;
        }
    }
}
```

Файл TimeSeriesClass.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Runtime.InteropServices;
using System.Windows;
using System.Data;

namespace DSS__Stock_Exchange
{
    static class TimeSeriesClass
    {
        private static List<double[]> dataArrays;
        public static DataTable dataTable { get; set; }

        static TimeSeriesClass()
```

```
{
    dataTable = new DataTable();
    dataTable.Columns.Add(new DataColumn("Номер ряду", typeof(int)));
    dataTable.Columns[0].AutoIncrement = true;
    dataTable.Columns[0].AutoIncrementSeed = 1;
    dataTable.Columns.Add(new DataColumn("Назва ряду", typeof(string)));
    dataTable.Columns.Add(new DataColumn("Довжина ряду", typeof(string)));

    dataArrays = new List<double[]>();
}

public static void AddDataAr(double[] newDataAr, string nameNewAr)
{
    while (IndexRow(nameNewAr) != -1)
    {
        nameNewAr += "_1";
    }
    double[] tmp = new double[newDataAr.Length];
    for (int i = 0; i < tmp.Length; i++)
    {
        tmp[i] = newDataAr[i];
    }
    dataArrays.Add(tmp);

    string[] s = new string[3];
    s[1] = nameNewAr;
    s[2] = tmp.Length.ToString();
    DataRow row = dataTable.NewRow();
    row.ItemArray = s;
    dataTable.Rows.Add(row);
}

public static void AddDataAr(int sizeNewDataAr, string nameNewAr)
{
    while (IndexRow(nameNewAr) != -1)
    {
        nameNewAr += "_1";
    }
    double[] tmp = new double[sizeNewDataAr];
    for (int i = 0; i < tmp.Length; i++)
    {
        tmp[i] = double.NaN;
    }
    dataArrays.Add(tmp);

    string[] s = new string[3];
    s[1] = nameNewAr;
    s[2] = tmp.Length.ToString();
    DataRow row = dataTable.NewRow();
    row.ItemArray = s;
    dataTable.Rows.Add(row);
}

public static void AddElements(int numberRow, int count)
{
    if (numberRow < dataArrays.Count)
    {
        double[] tmp = new double[dataArrays[numberRow].Length + count];
        for (int i = 0; i < tmp.Length; i++)
        {
            if (i < dataArrays[numberRow].Length) tmp[i] = dataArrays[numberRow][i];
            else tmp[i] = double.NaN;
        }
        dataArrays[numberRow] = tmp;

        dataTable.Rows[numberRow].ItemArray[2] = tmp.Length.ToString();
    }
}

public static void AddElements(string nameRow, int count)
{
    int numberRow = IndexRow(nameRow);
    if (numberRow >= 0)
    {
        AddElements(numberRow, count);
    }
```

```
}

public static void ReplaceDataAr(int numberRow,double[] newRow)
{
    if (numberRow < dataArrays.Count)
    {
        double[] tmp = new double[newRow.Length];
        for (int i = 0; i < tmp.Length; i++) tmp[i] = newRow[i];
        dataArrays[numberRow] = tmp;
        string[] s=new string[3];
        s[0] = dataTable.Rows[numberRow].ItemArray[0].ToString();
        s[1] = dataTable.Rows[numberRow].ItemArray[1].ToString();
        s[2] = tmp.Length.ToString();
        dataTable.Rows[numberRow].ItemArray = s;
    }
}

public static void DelDataAr(int numberRow)
{
    if (numberRow < dataArrays.Count)
    {
        dataArrays.RemoveAt(numberRow);
        dataTable.Rows.RemoveAt(numberRow);
    }
}

public static void DelDataAr(string nameRow)
{
    DelDataAr(IndexRow(nameRow));
}

public static void DelElement(int numberRow, int numberEl)
{
    if (numberRow<dataArrays.Count)
        if (numberEl < dataArrays[numberRow].Length)
        {
            double[] tmp = new double[dataArrays[numberRow].Length - 1];
            for (int i = 0; i < tmp.Length; i++)
            {
                if (i < numberEl)
                {
                    tmp[i] = dataArrays[numberRow][i];
                }
                if (i > numberEl)
                {
                    tmp[i] = dataArrays[numberRow][i+1];
                }
            }
            dataArrays[numberRow] = tmp;

            dataTable.Rows[numberRow].ItemArray[2] = tmp.Length.ToString();
        }
}

public static void DelElement(string nameRow, int numberEl)
{
    int numberRow = IndexRow(nameRow);
    if (numberRow >= 0)
    {
        DelElement(numberRow, numberEl);
    }
}

public static void EditElement(int numberRow, int numberEl, double newValue)
{
    if (numberRow<dataArrays.Count)
        if (numberEl < dataArrays[numberRow].Length)
        {
            dataArrays[numberRow][numberEl] = newValue;
        }
}

public static void EditElement(string nameRow, int numberEl, double newValue)
{
    EditElement(IndexRow(nameRow), numberEl, newValue);
}
```

```
public static void EditNameRow(int indexRow, string newNameRow)
{
    if (indexRow<dataTable.Rows.Count){
        string[] s = new string[3];
        s[0] = dataTable.Rows[indexRow].ItemArray[0].ToString();
        s[1] = newNameRow;
        s[2] = dataTable.Rows[indexRow].ItemArray[2].ToString();
        dataTable.Rows[indexRow].ItemArray = s;
    }
}

public static void EditNameRow(string nameRow, string newNameRow)
{
    EditNameRow(IndexRow(nameRow), newNameRow);
}

public static double Element(int numberRow, int numberEl)
{
    if (numberRow<dataArrays.Count)
        if (numberEl < dataArrays[numberRow].Length)
        {
            return dataArrays[numberRow][numberEl];
        }
    return double.NaN;
}

public static double Element(string nameRow, int numberEl)
{
    return Element(IndexRow(nameRow), numberEl);
}

public static int IndexRow(string nameRow)
{
    for (int i = 0; i < dataTable.Rows.Count; i++)
    {
        if (string.Equals(dataTable.Rows[i].ItemArray[1], nameRow))
        {
            return i;
        }
    }
    return -1;
}

public static int LenghtRow(int indexRow)
{
    if (indexRow < dataArrays.Count)
    {
        return dataArrays[indexRow].Length;
    }
    else return -1;
}

public static int LenghtRow(string nameRow)
{
    return LenghtRow(IndexRow(nameRow));
}

public static string NameRow(int indexRow)
{
    if (indexRow < dataArrays.Count)
    {
        return dataTable.Rows[indexRow].ItemArray[1].ToString();
    }
    else return null;
}

public static double[] ReturnRow(int indexRow)
{
    if (indexRow < dataArrays.Count)
    {
        double[] tmp = new double[dataArrays[indexRow].Length];
        for (int i = 0; i < tmp.Length; i++)
        {
            tmp[i] = dataArrays[indexRow][i];
        }
```

```
                return tmp;
            }
            else return null;
        }

        public static double[] ReturnRow(string nameRow)
        {
            return ReturnRow(IndexRow(nameRow));
        }

        public static int CountAllRows
        {
            get
            {
                return (int)dataTable.Rows[dataTable.Rows.Count-1].ItemArray[0];
            }
        }

        public static int CountRows
        {
            get
            {
                return dataTable.Rows.Count;
            }
        }

        public static string[] RowNames
        {
            get
            {
                string[] rez = new string[dataTable.Rows.Count];

                for (int i = 0; i < rez.Length; i++)
                {
                    rez[i] = dataTable.Rows[i].ItemArray[1].ToString();
                }

                return rez;
            }
        }
    }
}
```

Файл TimeSeriesShowWindow.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using System.Data;

namespace DSS__Stock_Exchange
{
    /// <summary>
    /// Логика взаимодействия для TimeSeriesShowWindow.xaml
    /// </summary>
    public partial class TimeSeriesShowWindow : Window
    {
        DataTable dataTable;
        int[] indexes;
        public TimeSeriesShowWindow(int[] indexes)
        {
            InitializeComponent();
            dataTable = new DataTable();
            dataTable.Columns.Add(new DataColumn("Номер", typeof(int)));
            dataTable.Columns[0].AutoIncrement = true;
```

```
            dataTable.Columns[0].AutoIncrementSeed = 1;
            int maxLength = 0;
            for (int i = 0; i < indexes.Length; i++)
            {
                dataTable.Columns.Add(new DataColumn(TimeSeriesClass.NameRow(indexes[i]), typeof(double)));
                if (maxLength < TimeSeriesClass.LenghtRow(indexes[i])) maxLength =
TimeSeriesClass.LenghtRow(indexes[i]);
            }

            for (int i = 0; i < maxLength; i++)
            {
                string[] tmp = new string[indexes.Length + 1];
                for (int j = 0; j < indexes.Length; j++)
                {
                    if (i < TimeSeriesClass.LenghtRow(indexes[j]))
                    {
                        tmp[j + 1] = TimeSeriesClass.Element(indexes[j], i).ToString();
                    }
                    else
                    {
                        tmp[j + 1] = double.NaN.ToString();
                    }
                }
                DataRow row = dataTable.NewRow();
                row.ItemArray = tmp;
                dataTable.Rows.Add(row);
            }

            TimeSeries.DataContext = dataTable.DefaultView;
            this.indexes = indexes;
        }

        private void Button_Click_Save_TimeSeries(object sender, RoutedEventArgs e)
        {
            try
            {
                for (int i = 0; i < indexes.Length; i++)
                {
                    double[] tmp = new double[dataTable.Rows.Count];
                    for (int j = 0; j < dataTable.Rows.Count; j++)
                    {
                        tmp[j] = double.Parse(dataTable.Rows[j].ItemArray[i+1].ToString());
                    }
                    TimeSeriesClass.ReplaceDataAr(indexes[i], tmp);
                    TimeSeriesClass.EditNameRow(indexes[i],TimeSeries.Columns[i+1].Header.ToString());
                }

            }
            catch
            {
                MessageBox.Show("Некоректні дані"," Помилка");
            }
        }

        private void Button_Click_Rename_TimeSeries(object sender, RoutedEventArgs e)
        {
            (new RenameSeriesWindow(TimeSeries.Columns)).Show();
        }

        private void Button_Click_Manipulate_TimeSeries(object sender, RoutedEventArgs e)
        {
            string[] names_row = new string[TimeSeries.Columns.Count - 1];
            for (int i = 0; i < names_row.Length; i++)
            {
                names_row[i] = TimeSeries.Columns[i + 1].Header.ToString();
            }
            (new TimeSeriesManipulateWindow(dataTable, names_row)).Show();
        }

        private void Button_Click_Show_TimeSeries(object sender, RoutedEventArgs e)
        {
            double[] x = new double[dataTable.Rows.Count];
            double[][] ys = new double[TimeSeries.Columns.Count - 1][];
            string[] names = new string[TimeSeries.Columns.Count - 1];
            for (int i = 0; i < x.Length; i++)
            {
```

```
                x[i] = double.Parse(dataTable.Rows[i].ItemArray[0].ToString());
            }
            for (int i = 0; i < names.Length; i++)
            {
                names[i] = TimeSeries.Columns[i+1].Header.ToString();
            }
            for (int i = 0; i < ys.Length; i++)
            {
                ys[i] = new double[dataTable.Rows.Count];
                for (int j = 0; j < ys[i].Length; j++)
                {
                    ys[i][j] = double.Parse(dataTable.Rows[j].ItemArray[i+1].ToString());
                }
            }

            (new ShowGraphicTimeSeriesWindow(x, ys, names)).Show();
        }

        private void Button_Click_Operations_TimeSeries(object sender, RoutedEventArgs e)
        {
            string[] names_row = new string[TimeSeries.Columns.Count - 1];
            for (int i = 0; i < names_row.Length; i++)
            {
                names_row[i] = TimeSeries.Columns[i + 1].Header.ToString();
            }
            (new TimeSeriesOperationWindow(dataTable, names_row)).Show();
        }
    }
}
```

ДОДАТОК Б ІЛЮСТРАТИВНІ МАТЕРІАЛИ ДЛЯ ДОПОВІДІ

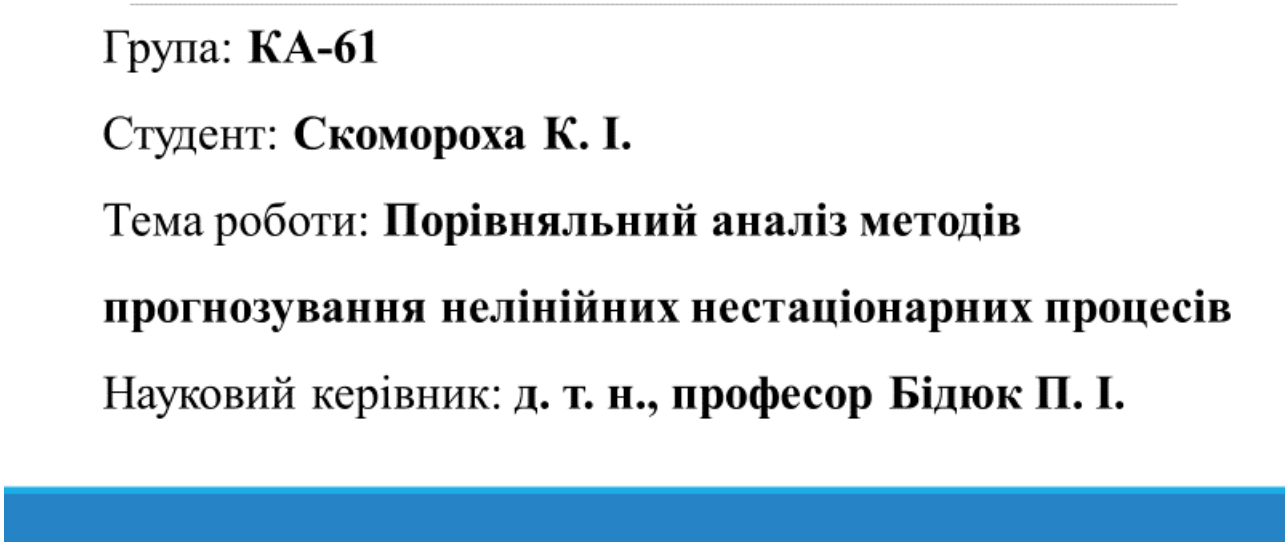


Рисунок Б.1

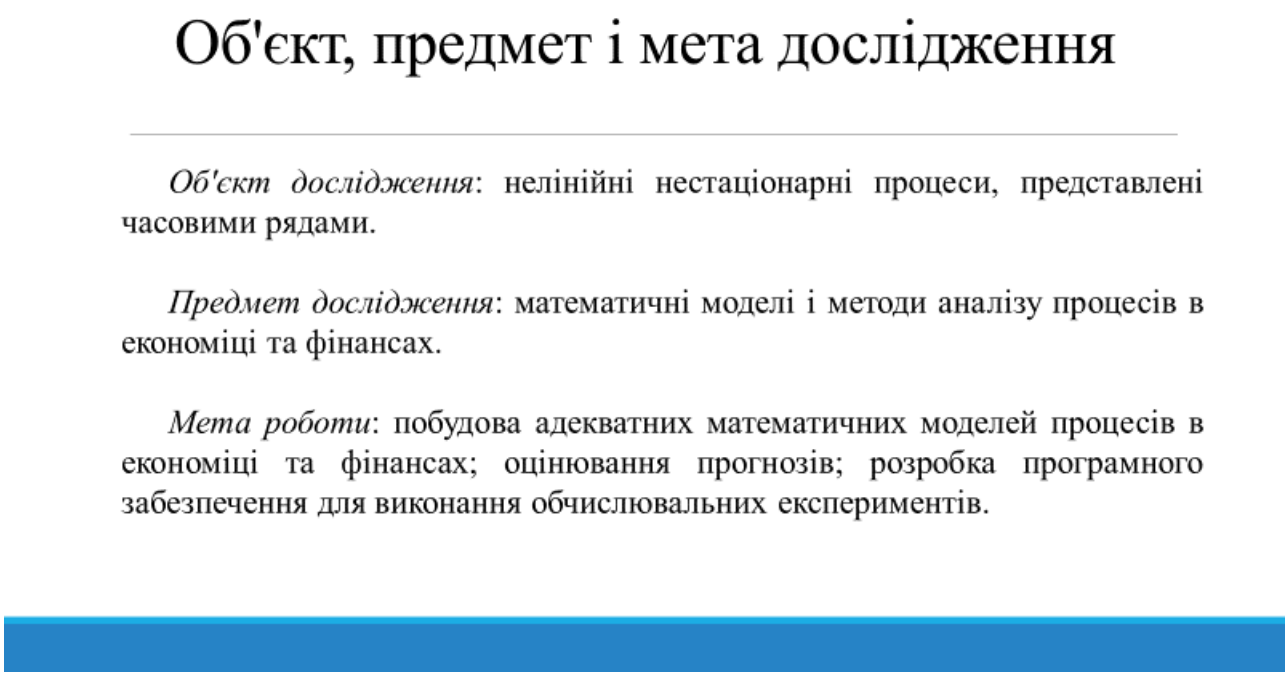


Рисунок Б.2

Актуальність дослідження

Прогнозування – найважливіший етап у процесі розробки стратегій розвитку, який дає можливість передбачити найбільш ймовірний розвиток подій, а також визначити, які дії приведуть до тих чи інших результатів.

Рисунок Б.3

Постановка задачі дослідження

- Виконати аналіз типів сучасних процесів.
- Аналіз методів і моделей прогнозування.
- Спроектувати і реалізувати СППР для моделювання і прогнозування фінансово-економічних нелінійних нестаціонарних процесів.
- Застосувати розроблену СППР до аналізу вибраних процесів – виконання обчислювальних експериментів.
- Виконати порівняльний аналіз результатів виконання обчислень за допомогою власного програмного забезпечення із уже існуючими.
- Виробити рекомендації стосовно варіантів майбутнього вдосконалення розробленої програми.

Рисунок Б.4

Існуючі системи для статистичної обробки

+ Потужність

+Візуалізація

+Продуктивність

+ Компактність

+ Доступність

Рисунок Б.5

Реалізовані моделі

Модель авторегресії з ковзним середнім АРКС(p, q):

$$y(k) = a_0 + \sum_{i=1}^{p} a_i y(k-i) + \sum_{j=1}^{q} b_j \varepsilon(k-j) + \varepsilon(k)$$

Інтегрована модель авторегресії з ковзним середнім АРІКС(p, d, q):

$$\Delta^d y(k) = a_0 + \sum_{i=1}^{p} a_i \Delta^d y(k-i) + \sum_{j=1}^{q} b_j \varepsilon(k-j) + \varepsilon(k)$$

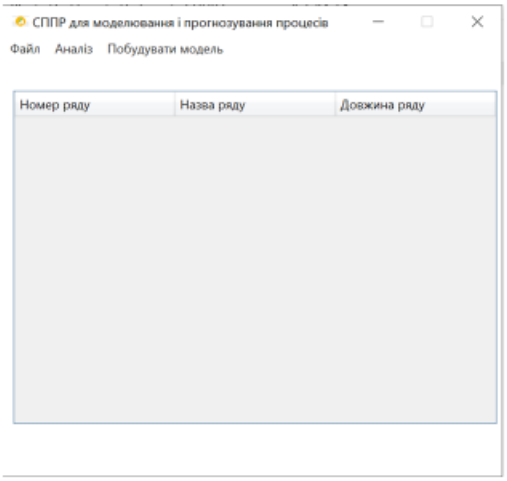
Рисунок Б.6

Реалізовані функції системи

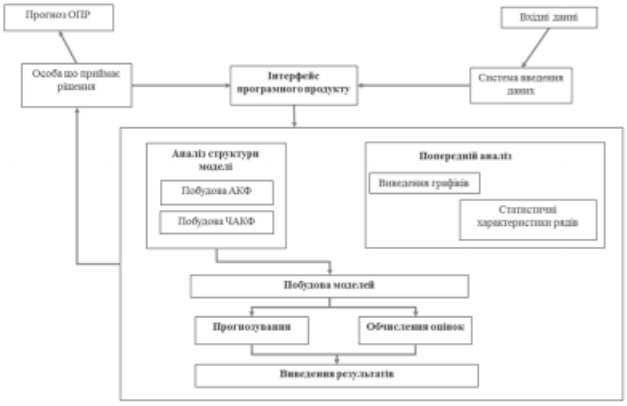
- ✓ Методи оцінювання параметрів моделі: метод найменших квадратів (МНК) і рекурентний метод найменших квадратів (РМНК);
- ✓ Кореляційні функції: автокореляційна функція (АКФ) і часткова автокореляційна функція (ЧАКФ); функція взаємної кореляції;
- ✓ Ковзне середнє: просте і експоненційне ковзне середнє.
- ✓ Візуалізація даних і результатів обчислень
- ✓ Статистичні характеристики для аналізу адекватності побудованих моделей (R2, DW) і оцінювання якості прогнозів (САПП, Тейл).

Рисунок Б.7

Функціональна схема СППР



Інтерфейс СППР

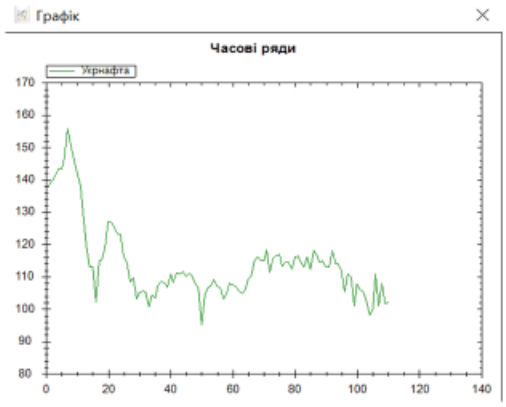


Функціональна схема

Рисунок Б.8

Результати моделювання і прогнозування цін акцій компанії «Укрнафта»

03.01.2018 – 15.06.2018



Вимір	Реальне значення	Розроблена СППР				Eviews	
		МНК		РМНК			
		Прогноз	Похибка	Прогноз	Похибка	Прогноз	Похибка
105	100.1	98.8984	1.6128	98.4872	1.6128	98.8984	1.6128
106	110.9	100.9178	9.9822	100.5585	10.3415	100.9178	9.9822
107	101	110.8316	9.8316	110.7267	9.7267	110.8316	9.8316
108	107.9	101.7439	6.1561	101.4058	6.4942	101.7439	6.1561
109	101.6	108.0778	6.4778	107.9022	6.3022	108.0778	6.4778
110	102	102.2947	0.2947	101.9707	0.0293	102.2947	0.2947
Сер. Похибка			5.7259		5.7511		5.7259
САПП		5.3843%		5.4695%		5.3843%	
САП		5.6573		5.7511		5.6573	

Рисунок Б.9

Результати моделювання і прогнозування цін акцій компанії «Укрнафта»

Модель процесу	Характеристики моделі			Характеристики прогнозу			
	R^2	$\sum e^2$	DW	СКП	САП	САПП	Тейла
АР(1)	0,87	1954,64	2,3	6,8	5,65	5,38%	0,03
АРКС(1,7)	0,71	793,57	1,99	3,58	2,74	2,61%	0,02
АР(3)+тренд	0,79	2345,14	2,16	8,31	7,85	7,23%	0,04

Прогноз на основі моделі АРКС(1,7) для цін акцій компанії «Укрнафта»

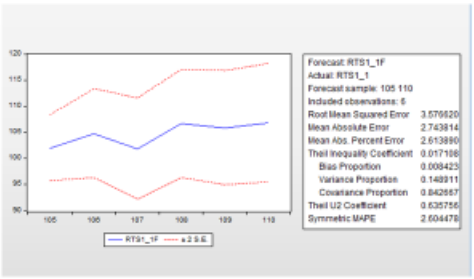
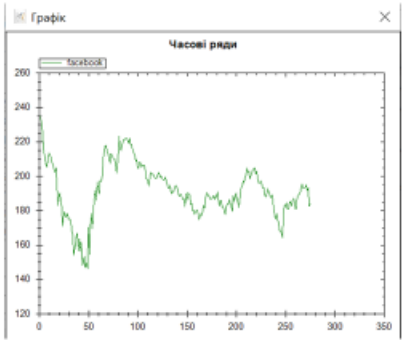


Рисунок Б.10

Результати моделювання і прогнозування цін акцій компанії «Facebook»

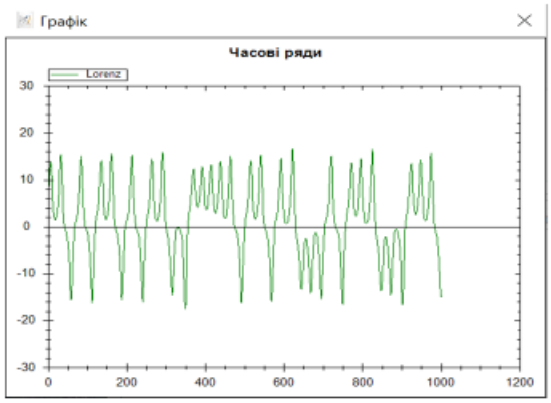
2019 рік



Модель процесу	Характеристики моделі			Характеристики прогнозу			
	R^2	$\sum e^2$	DW	СКП	САП	САПП	Тейла
АР(1)	0,9247	5150,58	2,33	4,56	3,09	1,65%	0,02
АР(10)	0,9266	4583,86	2,01	4,35	2,59	1,4%	0,01
АРКС(10,5)	0,9346	4083,75	2,02	2,29	2,29	1,24%	0,01
АР(3) +тренд	0,9296	4536,06	1,98	13,08	10,68	5,65%	0,03

Рисунок Б.11

Результати моделювання і прогнозування для процесу Лоренца



Модель процесу	Характеристики моделі			Характеристики прогнозу		
	R^2	$\sum e^2$	DW	СКП	САП	САПП
АР6	0,99998	1,186	1,135	7,98	6,29	144,36
АР8	0,99999	0,608	1,8875	7,8	6,13	136,95

$$x(k) = 0{,}001 + 5{,}53x(k-1) - 14{,}24x(k-2) + 22{,}52x(k-3) - 24{,}13x(k-4) +$$
$$+ 18{,}03x(k-5) - 9{,}19x(k-6) + 2{,}92x(k-7) - 0{,}44x(k-8)$$

Рисунок Б.12

Висновки за результатами дослідження

- Розроблена та програмно реалізована СППР для моделювання і прогнозування процесів на біржі. Система може бути використана також для моделювання і прогнозування процесів іншої природи.
- Створену СППР успішно тестовано на фактичних даних, які описують динаміку цін акцій компаній «Укрнафта», «Facebook» і нелінійний процес Лоренца.
- Встановлено, що точність моделювання та прогнозування, яка досягається за допомогою розробленої системи, збігається із результатами системи Eviews.
- Виконані обчислювальні експерименти свідчать, що лінійні моделі можуть апроксимувати нелінійні нестаціонарні процеси з високою адекватністю. Разом з тим, лінійні моделі не завжди забезпечують належну апроксимацію та обчислення прогнозів нелінійних процесів необхідної якості.

Рисунок Б.13

Шляхи подальшого розвитку результатів дослідження

- Розширення множини методів попередньої обробки даних – фільтрація, аналіз екстремальних значень, тестування на нелінійність та нестаціонарність і т. ін.
- Впровадження в систему альтернативних методів оцінювання параметрів моделей, що дасть можливість оцінювати моделі, нелінійні стосовно параметрів (наприклад, методу Монте-Карло для марковських ланцюгів).
- Реалізація функцій побудови альтернативних моделей досліджуваних процесів (наприклад, на основі інтелектуального аналізу даних).
- Автоматизація процесу оцінювання структури моделі, подальший розвиток інтерфейсу.
- Реалізація методів комбінування оцінок прогнозів з метою подальшого підвищення їх якості.

Рисунок Б.14

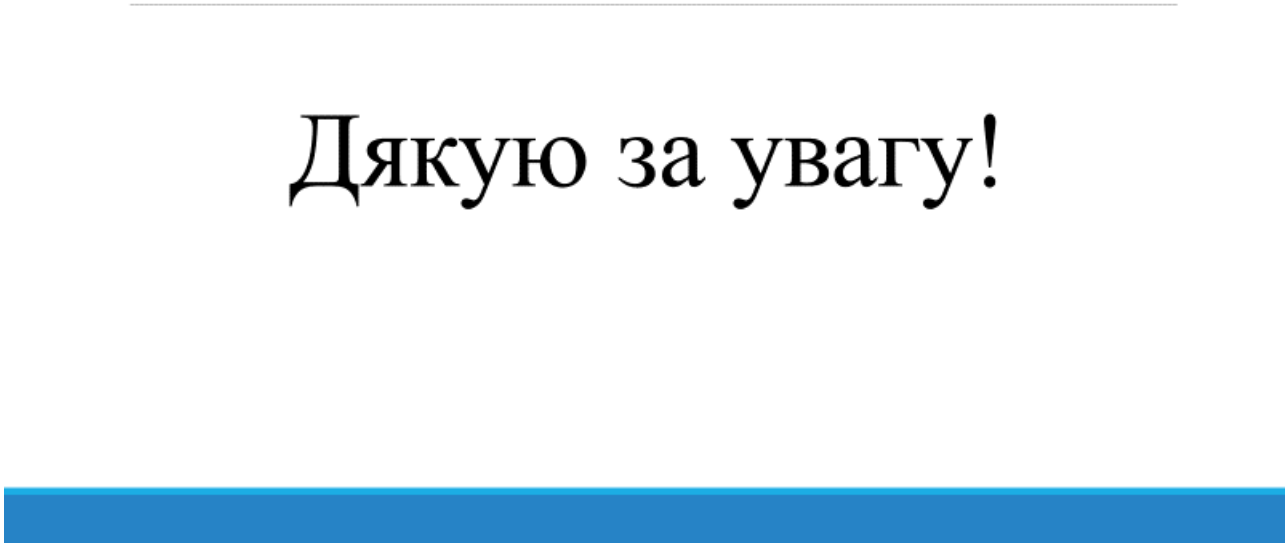


Рисунок Б.15